

PÉRIPHÉRIQUES ET FICHIERS SUR AMSTRAD CPC 464, 664 et 6128

DANIEL-JEAN DAVID

PÉRIPHÉRIQUES ET FICHIERS SUR AMSTRAD CPC 464, 664 et 6128

CONNAISSEZ-VOUS TOUTE LA COLLECTION AMSTRAD CHEZ P.S.I.?

Pour les Amstrad CPC 464, 664 et 6 128

Initiation :

— La découverte de l'Amstrad - Daniel-Jean David
— Exercices en Basic pour Amstrad - Maurice Charbit

Programmation BASIC :

— 102 programmes pour Amstrad - Jacques Deconchat
— Amstrad en famille - Jean-François Sehan
— Super jeux Amstrad - Jean-François Sehan
— Super générateur de caractères sur Amstrad - Jean-François Sehan
— Photographie sur Amstrad et Apple II - Pierrick Moigneau et Xavier de la Tullaye

Maîtrise du BASIC :

— Basic Amstrad 1 - Méthodes pratiques - Jacques Boisgontier et Bruno César
— Basic Amstrad 2 - Programmes - Jacques Boisgontier
— Basic plus 80 routines sur Amstrad - Michel Martin

Assembleur et Pascal :

— Assembleur de l'Amstrad - Marcel Henrot
— Turbo Pascal sur Amstrad - Pierre Brandeis et Frédéric Blanc

Système :

— Clefs pour Amstrad 1 - Système de base - Daniel Martin
— Clefs pour Amstrad 2 - Système disque - Daniel Martin et Philippe Jadoul
— Le livre de CP/M Plus sur Amstrad 6 128 et 8 256 - Yvon Dargery

A paraître :

— Amstrad en musique - Daniel Lemahieu
— Intelligence artificielle : langages et formes sur Amstrad - Thierry Lévy - Abegnoli et Olivier Magnan
— Création et animations graphiques sur Amstrad - Gilles Fouchard et Jean-Yves Corre
— Simulation et intelligence artificielle sur Amstrad - René Descamps
— Clefs pour Amstrad 8 256 - Eric Baumarti

Pour tout problème rencontré dans les ouvrages P.S.I.
vous pouvez nous contacter au numéro ci-dessous :

Numéro Vert/Appel Gratuit en France

05 21 22 01

(Composer tous les chiffres, même en région parisienne)

ISBN 2-86595-316-5

DANIEL-JEAN DAVID

PÉRIPHÉRIQUES ET FICHIERS SUR AMSTRAD CPC 464, 664 et 6128



ÉDITIONS DU PSI
1985

Daniel-Jean David enseigne l'informatique de gestion à l'Université de Paris-1 Panthéon Sorbonne.

Par ailleurs, il enseigne l'utilisation des microprocesseurs à l'E.N.S.A.M.

Ses sujets de recherche vont de l'informatique graphique aux techniques d'interface des microprocesseurs et des systèmes multiprocesseurs.

Spécialiste des microprocesseurs, il a donné de nombreux séminaires tant pour débutants qu'au niveau le plus élevé et il a écrit de nombreux articles dans les revues spécialisées.

SOMMAIRE

PRÉSENTATION

Faisant suite à *La découverte de l'Amstrad* qui était destiné à accompagner les premiers pas de l'utilisateur de ce micro-ordinateur, ce livre a pour but d'approfondir vos connaissances en BASIC pour tout ce qui concerne la manipulation des fichiers et des périphériques.

Cette connaissance est probablement la plus importante du point de vue des applications : toute application n'a-t-elle pas, à un moment ou à un autre, à manipuler des données groupées en fichiers ?

Ce livre est valable pour les trois modèles d'Amstrad actuellement sur le marché : le 464 à magnétocassette et le 664 ou le 6128 à disquette, sachant que les cassettes et les disquettes se manipulent de façons assez semblables et que le 464 peut recevoir une unité de disquettes externe tandis que l'on peut connecter un magnétophone à cassettes ordinaire à l'extérieur du 664/6128.

Le premier chapitre donne l'art et la manière de stocker puis de retrouver des données sur cassettes. C'est une introduction indispensable à la suite du livre pour trois raisons :

1. – la cassette est le périphérique magnétique le moins cher ;
2. – la forme des instructions BASIC d'utilisation de la cassette est conservée pour les autres périphériques ;
3. – les avantages et les inconvénients de la cassette seront évalués du point de vue des applications, ce qui introduira la nécessité des autres périphériques.

Le chapitre suivant donne le mode d'emploi des imprimantes connectables à l'Amstrad par l'intermédiaire de l'interface imprimante. L'imprimante est manipulable par des ordres BASIC standard.

Le chapitre 3 traite des disques connectables à l'Amstrad 464, ou déjà incorporés dans le 664/6128. Ils sont utilisables en accès séquentiel à l'aide d'ordres BASIC standard. Pour les utiliser en accès direct, nous donnons le moyen de pallier l'absence d'instructions adéquates du BASIC de l'Amstrad.

Mais ranger des données sur bande ou sur disque ne suffit pas. Il faut, au préalable, organiser les données dont on a besoin pour une application. Le chapitre 4 a pour but de fournir des notions sur les structures et les organisations de données les plus utilisées. Il veut être une modeste introduction aux bases de données.

Le chapitre 5 décrit enfin l'utilisation des autres périphériques que possède l'Amstrad, ou qui sont annoncés.

On trouvera au cours du chapitre 3, et un peu au chapitre 1, le listing commenté d'un système de gestion de données sur disque pouvant servir à vos besoins personnels ou à ceux d'une P.M.E.

Enfin, l'ouvrage se termine par des annexes : messages d'erreur, questions et réponses, solutions des exercices.

CHAPITRE 1
DONNÉES SUR CASSETTES

RAPPEL

Vous vous rappelez *(cf. La découverte de l'Amstrad)* les instructions qui permettent de sauvegarder, un programme sur cassette, puis de le retrouver ? Cela montre que la cassette n'est pas si difficile à utiliser. Nous allons revoir ces instructions puis les généraliser et les étendre.

Pour sauver un programme, on utilise SAVE.

Pour recharger le programme, on utilise LOAD "nom".

Note pour les possesseurs de 664/6128

Si vous désirez suivre les exemples de ce chapitre exactement sur cassette, vous devez connecter un magnétophone ordinaire à la prise DIN prévue pour cela à l'arrière de votre 664 – sur le 6128, le connecteur est sur le côté gauche – (schéma dans la notice constructeur) et vous devez rediriger les entrées-sorties sur cassette à l'aide de la commande ¦TAPE (la barre ¦ s'obtient par SHIFT @).

Si vous ne le faites pas, les opérations se dérouleront sur disque exactement comme si c'était une cassette, sauf que vous n'aurez pas les messages demandant d'appuyer sur telle ou telle touche du magnétophone.

Inversement, si vous avez un 464 muni d'une unité de disquettes externe, vous pouvez l'utiliser au lieu de la cassette, à condition de taper la commande ¦DISC.

LE SYSTÈME D'EXPLOITATION

Dans toute la suite, nous ferons beaucoup d'allusions à ce que nous appellerons "le système". Il s'agit du *système d'exploitation* de l'ordinateur. Nous l'avons défini succinctement dans *La découverte de l'Amstrad.* Rappelons qu'il s'agit de l'ensemble des programmes qui sont livrés avec l'ordinateur et qui aident, ou même sont indispensables, à son utilisation.

La composante principale du système d'exploitation utilisée dans le livre précédent était l'interpréteur BASIC. Le système peut contenir d'autres interpréteurs de langages. Ainsi, vous pouvez vous procurer un moniteur d'aide à l'introduction de programmes en langage-machine ; si vous disposez de disques, la disquette système qui vous est livrée contient un interpréteur du langage Logo qui est tout à fait intéressant.

Mais une autre catégorie de routines du système d'exploitation est très importante et elle le sera d'autant plus dans ce livre consacré aux périphériques : ce sont les routines de gestion des entrées-sorties.

Si nous pouvons taper au clavier, c'est parce qu'il existe une routine du système d'exploitation qui va voir si quelqu'un a tapé quelque chose au clavier.

Si nous obtenons des résultats à l'écran, c'est parce qu'il existe un programme du système d'exploitation qui gère l'affichage... et ainsi de suite pour tous les périphériques.

En fait, plus que l'étude du comportement physique des périphériques, ce livre est l'étude des commandes qu'il faut envoyer au système d'exploitation pour obtenir les actions que nous souhaitons d'un périphérique déterminé.

Sur Amstrad, le système d'exploitation est en plusieurs morceaux : la gestion des périphériques autres que les disques est en ROM avec BASIC. Quant au système de gestion disque (on dit système d'exploitation disque ou SED, en anglais DOS - Amstrad l'appelle AMSDOS), il est en ROM avec le BASIC 1.1 sur 664/6128, tandis qu'il est en ROM externe sur 464. De plus, les disques Amstrad sont livrés avec un autre système d'exploitation disque, plus perfectionné que AMSDOS, CP/M, dont le principal avantage est qu'il existe beaucoup de logiciels commerciaux qui tournent avec lui. Dans ce livre, nous ne l'utiliserons que pour quelques commandes au chapitre 3 : son utilisation est surtout tributaire des logiciels sous CP/M que vous aurez achetés.

COMPLÉMENTS SUR LES INSTRUCTIONS DE SAUVEGARDE

SAVE

La forme complète de l'instruction SAVE est :

SAVE "nom",type,adresses

où nom est le nom donné au fichier ;
type est une lettre qui définit le type du fichier constitué, les valeurs possibles étant :

– rien : fichier programme ordinaire ;

– P : fichier programme protégé ; vous ne pourrez pas faire LOAD sur ce programme, mais seulement RUN ou CHAIN. Attention, il faut faire une première sauvegarde non protégée, pour garder par devers vous une version manipulable du programme, si vous voulez diffuser sous forme protégée un programme que vous avez écrit ;

– A : fichier ASCII ; le fichier est chargeable par LOAD comme tout fichier programme, mais aussi lisible par les instructions INPUT# comme un fichier de données ;

– B : fichier binaire, image exacte d'une zone mémoire. Il faut dans ce cas la présence des paramètres adresses a1,a2,a3 : a1 est l'adresse de début, a2 est le nombre d'octets à écrire et a3 (facultatif) est l'adresse de point d'entrée. Cela sert surtout à sauvegarder la mémoire d'écran pour conserver une image SAVE "ECRAN",B,&C000,&4000. Cet exemple montre que les adresses peuvent être spécifiées en hexadécimal.

Les noms

Sur cassette, le nom est une chaîne de caractères ordinaires (16 caractères au maximum – vous pouvez en donner plus, mais le système n'en retiendra que 16). Les majuscules et les minuscules sont considérées comme équivalentes.

Sur disque, le nom (8 caractères au maximum) peut être suivi d'une extension facultative '. et 3 caractères'. L'extension est arbitraire mais elle joue un rôle intéressant pour vous rappeler ce qu'il y a dans le fichier. *Ex.*–.BAS pour un programme BASIC. Le nom peut être précédé de A: ou B: pour dire sur

quelle unité de disque est le fichier. Nous n'en parlerons plus supposant que, pour le moment, vos finances ne vous ont permis que l'achat d'une unité de disques : c'est alors toujours A.

Certaines extensions ont un sens particulier comme .BAK pour un fichier copie de sécurité, ou .COM pour un fichier exécutable sous CP/M. Le SAVE d'un programme BASIC donne automatiquement l'extension .BAS.

Question : que se passe-t-il si, lors du SAVE, le fichier existe déjà ? – Le système crée le nouveau fichier sous le nom spécifié tout en conservant l'ancien sous le même nom, mais avec l'extension .BAK. Bien sûr, il n'y a rien de tout cela sur cassette : si vous écrivez un nouveau fichier alors que la cassette est positionnée à l'emplacement d'un ancien fichier, l'ancien fichier est écrasé, quelque soit le nom, d'ailleurs. Si, au contraire, vous êtes positionné sur une partie vide de la cassette, alors le fichier est créé sans en écraser un autre. Il peut alors y avoir deux fichiers de même nom sur la cassette : c'est le premier rencontré qui sera chargé lorsque vous le demanderez.

Sur cassette, le nom peut être précédé d'un point d'exclamation. Ce '!' ne sera pas considéré comme faisant partie du nom, il aura pour effet de supprimer le message vous demandant d'appuyer sur le(s) bouton(s) voulu(s) du magnétophone et l'attente que vous l'ayez fait (vous signalez que c'est fini en appuyant sur une touche quelconque du clavier). Il faut, bien sûr, que la cassette soit bien positionnée et les boutons dans l'état convenable. Cela se justifie plus pour des opérations de lecture ou pour des fichiers de données. Pour une sauvegarde, c'est assez imprudent : si le magnétophone était en lecture ou même à l'arrêt, le système croirait la sauvegarde effectuée alors qu'elle ne le serait évidemment pas !

Vérification de la sauvegarde

L'Amstrad n'a pas d'instruction spécifique pour vérifier que la sauvegarde s'est bien déroulée, mais si vous donnez une commande CAT en ayant réembobiné la bande pour qu'elle soit positionnée juste devant le programme que vous venez de sauver, le système va citer ce programme et il marquera Ok si le programme est correctement écrit. Il est prudent d'effectuer cette opération, car si vous décelez que le programme n'est pas lisible pendant qu'il est encore en mémoire, il vous reste la possibilité de le sauver sur une autre cassette.

Les deux vitesses d'écriture

Si vous effectuez une commande SPEED WRITE 1, toutes les opérations d'écriture sur cassette se feront à vitesse double de la normale. On revient à la vitesse normale par la commande SPEED WRITE 0. De toutes façons, les

fonctions de lecture savent s'adapter à la vitesse à laquelle un fichier avait été écrit. Bien sûr, la commande est inopérante sur disque.

La grande vitesse a un intérêt évident, mais elle exige des cassettes de qualité correcte et elle est moins apte à absorber les parasites et déréglages du magnétophone. Il est donc plus prudent de se limiter à la vitesse normale.

LOAD

L'instruction de chargement est de la forme :

LOAD "nom"

Elle s'applique à tout fichier constitué par SAVE, sauf de type protégé. Pour un fichier binaire, on peut ajouter un paramètre adresse, si l'on désire que l'adresse de chargement ne soit pas celle de la sauvegarde.

Exécution automatique

L'instruction RUN "nom" est équivalente à LOAD puis RUN, c'est-à-dire qu'elle fait charger le programme spécifié puis en démarre aussitôt l'exécution.

Une différence par rapport à LOAD est que RUN peut s'appliquer à un programme protégé (sauvé avec l'option P). Avec LOAD, un tel programme est chargé, mais détruit aussitôt après, de sorte que vous ne pouvez le lister ou le copier. Avec RUN, un tel programme sera chargé et exécuté.

Fusion des programmes

Lorsque vous chargez un programme par LOAD, s'il y avait déjà un programme en mémoire, celui-ci est effacé pour être remplacé par le programme que vous chargez.

Si vous employez la commande MERGE "nom" au contraire, le nouveau programme est fusionné avec l'ancien. Tout se passe comme si le programme lu était tapé au clavier : les instructions de numéros différents des numéros déjà présents sont ajoutées ; les instructions de numéro égal à un numéro déjà présent remplacent leurs homologues.

Exemple

Pour vous en persuader, faites le programme suivant :

```
10 ?"TOTO"
20 ?"LOLO"
30 ?"ZOZO"
```

Faites SAVE "X" puis NEW. Entrez maintenant :

```
5  ?"LILI"
10 ?"ZIZI"
15 ?"NINI"
```

Faites maintenant MERGE "X". Le programme définitif est :

```
5  ?"LILI"
10 ?"TOTO"
15 ?"NINI"
20 ?"LOLO"
30 ?"ZOZO"
```

Cela permet des applications très intéressantes. Vous pouvez notamment vous constituer des bibliothèques de sous-programmes que vous pourrez incorporer à tous les programmes que vous voudrez, pourvu que les numéros de lignes soient bien choisis.

LES OVERLAYS

Il est parfaitement possible de placer une instruction LOAD dans un programme, par exemple :

```
1000 LOAD "SUITE"
```

Cette instruction sera la dernière instruction exécutée du programme où elle se trouve : en effet, elle fait charger le programme "SUITE" à partir de la cassette ; celui-ci vient remplacer en mémoire le programme présent et l'on se retrouve en mode direct.

Le LOAD ne produit que le chargement du second programme, ce qui n'a pas grand intérêt. Si l'on veut que le programme "SUITE" s'exécute, il faut taper RUN ou alors faire :

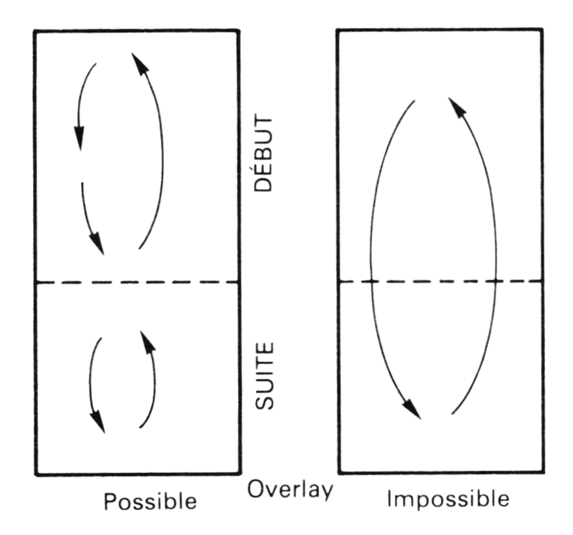
1000 RUN "SUITE"

au lieu du LOAD.

En principe, le programme "SUITE" sera sur la cassette juste après le programme "DEBUT", et si la touche **PLAY** du magnétophone a été laissée enfoncée, l'utilisateur n'a plus qu'à taper sur une touche quelconque de l'ordinateur en réponse au message du système, pour que le chargement et l'exécution du second programme s'effectuent. On peut même supprimer le message et l'intervention de l'utilisateur si l'on fait précéder le nom d'un point d'exclamation : 1000 RUN "!SUITE". Bien sûr, si l'on est sur disque, il n'y a ni message, ni besoin d'appuyer sur une touche, ni problème de position pour le programme. On a donc un enchaînement automatique des deux programmes.

L'intérêt de ceci est évident : supposons que nous ayons un programme trop long pour tenir dans la mémoire de l'Amstrad (c'est peu probable, car avec plus de 40 000 octets disponibles, on a déjà de quoi faire !). On peut alors le scinder en deux parties "DEBUT" et "SUITE" qui s'enchaîneront si "DEBUT" se termine par un RUN "SUITE".

Il est impératif pour que ceci fonctionne qu'il n'y ait pas de sauts (GOTO ou IF) entre la première et la seconde partie *(cf. figure ci-dessous).*



Cette technique s'appelle *overlay* ("recouvrement" en anglais : en effet SUITE vient recouvrir DEBUT en mémoire).

La généralisation à toute une série de programmes qui s'enchaînent l'un à l'autre est possible. Une structure en boucle où le dernier recharge le premier et ainsi de suite est imaginable : il faut à ce moment un réembobinage manuel du magnétophone ; c'est plus commode sur disque.

Conservation des variables – CHAIN

Il y a un problème qui rend nécessaire une autre instruction que RUN : avec RUN toutes les variables sont perdues entre la première et la seconde partie (puisqu'on fait RUN, précisément).

Si, au lieu de RUN, vous écrivez :

```
1000 CHAIN "SUITE"
```

alors les variables seront transmises de la première à la seconde partie.

Il y a même une variante encore plus élaborée qui permet de conserver en mémoire une partie du programme de départ. On écrit par exemple :

```
1000 CHAIN MERGE "SUITE",10050,DELETE 10000–
```

qui signifie "garder les instructions jusqu'à 10000, supprimer celles de numéros >=10000, charger "SUITE" avec fusion (on suppose que SUITE n'a que des instructions >10000) et lancer à partir de 10050".

Ceci permet une structure où un segment racine appelle successivement différents segments subsidiaires qui se recouvrent en mémoire. Cette possibilité permet à un Amstrad de traiter des problèmes extrêmement complexes, mais elle est surtout pratique avec les disques et, répétons-le, on n'en a pas très souvent besoin, maintenant que les ordinateurs familiaux ont des tailles mémoire importantes.

FICHIERS DE DONNÉES SUR CASSETTE

Une autre manière d'augmenter la taille mémoire apparente de votre ordinateur serait de pouvoir ranger des données sur la cassette et de les récupérer plus tard. La place mémoire laissée libre par ces données serait alors affectée entre temps à d'autres données. En définitive, cela vous permettrait de traiter plus de données.

L'Amstrad le permet parfaitement, mis à part les inconvénients dus à la lenteur de la cassette. On distingue les instructions de gestion des fichiers (et de gestion des noms) et celles de transfert proprement dit des informations. Ces dernières sont très semblables aux instructions INPUT et PRINT que nous connaissons déjà.

Toute opération d'utilisation d'un fichier sur cassette est la succession de trois phases :

1. – définition et préparation du fichier : c'est l'instruction OPEN ;

2. – transferts de données proprement dits : instructions INPUT# pour lire, PRINT# pour écrire ;

3. – terminaison des opérations : c'est l'instruction CLOSE.

L'instruction OPEN

Un fichier présent sur la cassette ou la disquette est caractérisé par plusieurs paramètres : le périphérique concerné ; l'action qui va être réalisée : lecture ou écriture (le système doit en être prévenu) et le nom du fichier.

Il serait très fastidieux d'avoir à répéter ces paramètres à chaque fois qu'une lecture ou une écriture doit être effectuée sur le fichier. C'est pourquoi on utilise un *résumé* de ces informations pour désigner le fichier dans les ordres de transfert de données.

Ce résumé a la forme d'un simple nombre entier appelé *numéro logique de fichier.* C'est ce numéro qui est référencé dans chaque instruction d'utilisation du fichier.

Les numéros possibles vont de 0 à 9. Le numéro détermine à lui seul le périphérique concerné : de 0 à 7, il s'agit d'une fenêtre d'écran ; 8 désigne l'imprimante ; 9 désigne un fichier disque ou cassette.

Le choix entre disque ou cassette est fait par une instruction ¦TAPE ou ¦DISC ; sinon, par défaut, c'est la cassette qui est utilisée sur 464, et la disquette sur 664/6128.

Lorsque le périphérique est 9 (cassette ou disquette), il y a les autres paramètres (lecture/écriture et nom de fichier) à préciser : c'est l'instruction OPEN qui fournit ces paramètres une fois pour toutes.

Elle est de la forme :

OPENIN nom
ou OPENOUT nom

où IN signifie "lecture", OUT signifie "écriture" et le nom est une chaîne de caractères qui a exactement la même forme que ce que nous avons vu pour

les sauvegardes de programmes. Le nom peut être spécifié sous forme de chaîne entre guillemets (*ex.* "MONFICH") ou sous forme de variable alphanumérique (NOM$="MONFICH": OPENIN NOM$). Le numéro logique n'a pas à être rappelé puisque c'est toujours 9 vu que les paramètres supplémentaires ne jouent que pour disques ou cassettes.

Voici quelques exemples avec leur interprétation :

```
10 ¦TAPE : OPENIN "MONFICH"
```

fichier sur cassette, en lecture, de nom MONFICH.

```
20 ¦TAPE : A$="!ZOZO" : OPENOUT A$
```

fichier sur cassette, en écriture, nom ZOZO spécifié sous forme de variable, suppression des messages de manipulation du magnétophone.

```
30 ¦DISC : N$="MONFIC": E$=".DAT"
40 OPENIN N$+E$
```

fichier en lecture sur disquette, nom MONFIC.DAT (remarquez l'extension et la concaténation).

Exercice 1.1. – Retrouvez un fichier en lecture sur le magnétophone. Le fait qu'on utilise le magnétophone doit être indiqué au dernier moment par l'opérateur.

Fonctionnement de l'instruction OPEN

L'instruction OPEN gère essentiellement l'enregistrement d'en-tête du fichier qui contient le nom et diverses informations. Le comportement est différent en lecture et en écriture.

En lecture, comme dans un LOAD, le système attend de trouver l'enregistrement d'en-tête. Il est alors possible de lire le premier enregistrement de données.

En écriture, comme dans un SAVE, le système crée l'enregistrement d'entête. Les enregistrements de données peuvent alors être écrits successivement.

Un autre rôle de l'instruction OPEN est de demander à l'utilisateur d'enfoncer les touches convenables du magnétophone (et d'appuyer sur une touche quelconque du clavier lorsque c'est fait). Une fois les touches du magnétophone enfoncées, elles doivent le rester tout le temps d'utilisation. Rappelons que ces messages sont supprimés si l'on fait précéder le nom de fichier du caractère '!'.

En écriture, il faut s'assurer que le magnétophone est positionné sur une zone vierge de la cassette ; sinon, des informations précieuses peuvent être perdues.

Rappelons le conseil d'éviter de mettre trop de fichiers ou de programmes sur une même cassette : sinon, vous perdrez du temps en recherche des fichiers.

Sur disquette, il n'y a, bien sûr, pas de messages ni de touches à appuyer. Si, en écriture, vous demandez un nom qui existe déjà, tout se passera comme pour SAVE : votre nouveau fichier remplacera l'ancien, mais l'ancien sera conservé avec l'extension .BAK.

Si, en lecture, vous spécifiez un fichier qui n'existe pas, le système fera défiler toute la cassette. Sur disquette, il y aura le message d'erreur 'nom' not found (non trouvé).

Notons dès maintenant – nous y reviendrons – une sérieuse limitation du BASIC de l'Amstrad : il ne peut, à un instant donné, n'y avoir qu'un fichier ouvert sur périphérique magnétique, même si vous avez à la fois disque et cassette ou deux unités de disque, puisqu'il n'y a qu'un numéro possible, le 9.

Instructions INPUT# et PRINT#

Une fois le fichier ouvert, il faut effectuer les transferts proprement dits des informations. En lecture, c'est l'instruction INPUT# très analogue à INPUT que nous connaissons déjà, qui est utilisée. En écriture, c'est l'instruction PRINT#, analogue à PRINT, qui est utilisée.

Les instructions de transfert sont de la forme :

```
INPUT#9,liste
PRINT#9,liste
```

où le 9 est le numéro logique de fichier ; nous avons déjà dit que, sur l'Amstrad, c'est toujours 9 pour les cassettes ou disquettes.

'liste' est la suite des noms des variables dont les valeurs doivent être lues ou écrites.

Dans INPUT#, les différents noms sont séparés par des virgules. Dans PRINT#, ils sont séparés soit par une virgule, soit par un point-virgule.

Les virgules entraînent une tabulation (passage à la zone suivante : la largeur de zone est définie par l'instruction ZONE) tandis que les ';' font coller les écritures. Le comportement est exactement le même que sur l'écran avec PRINT.

Exemple

10 INPUT#9,A	lit le nombre A ;
20 INPUT#9,X,Y,A$,B$	lit successivement les nombres X, Y puis les chaînes de caractères A$ et B$;
30 PRINT#9,A;B 40 PRINT#9,A$,B$;A 50 PRINT#9, A$;B$	écrit A et B serrés.

Exercice 1.2. – Le programme ci-dessous crée un fichier sur lequel il inscrit les carrés des cinquante premiers nombres. Ecrire un programme qui relit ce fichier et l'imprime sur l'écran :

```
10 REM CREATION D'UN FICHIER
20 OPENOUT "CARRES"
30 FOR I=1 TO 50 : K=I^2
40 PRINT#9,K : NEXT I
50 CLOSEOUT ; REM VOIR CI-DESSOUS
60 END
```

Pour faire cet exercice avec profit, nous vous conseillons de prendre une cassette vierge sur laquelle s'inscrira le fichier de données.

Nous donnons maintenant, sans les justifier pour le moment, deux règles qui découlent du bon sens et vous éviteront certains déboires. N'oublions pas qu'en informatique, si le système a la moindre chance de ne pas marcher, il la saisira et il ne marchera effectivement pas.

– **Règle 1** : assurez-vous que vous ne cherchez à lire que ce que vous avez écrit. Il faut donc que la liste des variables en lecture soit identique à celle de l'écriture : même nombre de variables, et même type, numérique ou alphanumérique.

– **Règle 2** : vous avez un meilleur contrôle en écrivant (et en lisant) les variables une par une : PRINT#9,A. Vous avez un contrôle encore meilleur en utilisant des variables alphanumériques : PRINT#9,A$ (pour des nombres utiliser STR$ et VAL).

Un dernier point. – Vous savez que l'abréviation de PRINT est '?'. Et bien, vous pouvez écrire '?#' pour PRINT#.

L'instruction CLOSE

Une fois que tous les transferts d'informations ont été effectués avec le fichier, il faut le fermer comme tout employé de bureau ferme les fichiers qu'il a utilisés quand il a terminé de les consulter.

C'est l'objet de l'instruction CLOSE qui s'écrit :

CLOSEIN

pour un fichier en lecture et

CLOSEOUT

pour un fichier en écriture.

Là encore, le numéro logique n'est pas rappelé puisque c'est toujours 9. Cette limitation à un seul fichier accessible à un instant donné rend l'instruction CLOSE d'autant plus indispensable : on peut utiliser plusieurs fichiers dans un programme, mais pas simultanément et il faut fermer le fichier qu'on vient d'utiliser avant de pouvoir utiliser le suivant.

Une autre utilité fondamentale de l'instruction CLOSE est due à l'organisation des données en blocs.

Le tampon

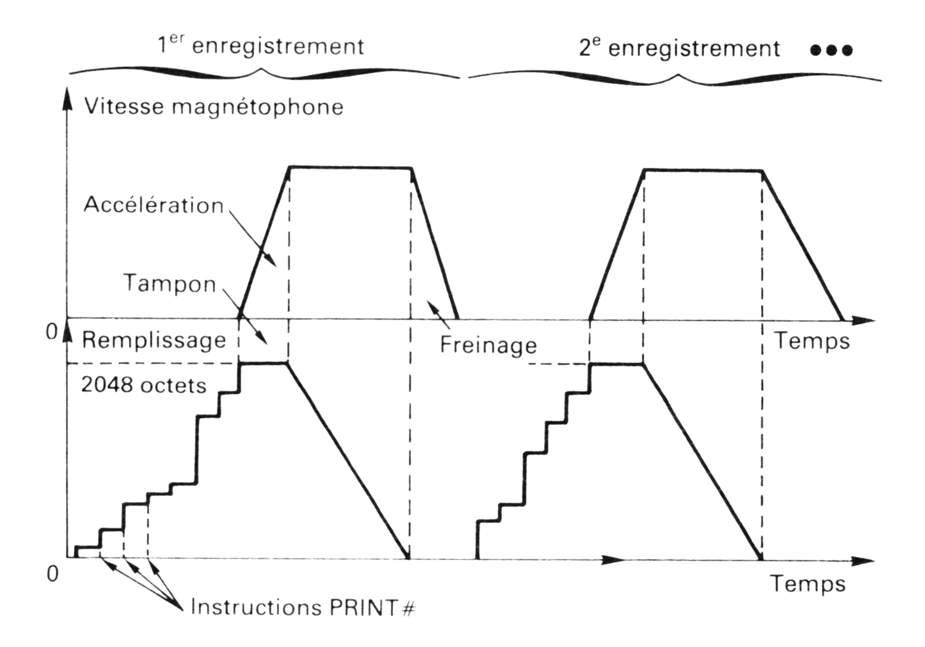
Lorsque l'instruction PRINT#9,A est exécutée, croyez-vous que le magnétophone démarre et que le nombre A est écrit immédiatement sur la bande ? Pas du tout !

En effet, une écriture ne peut avoir lieu que lorque la bande a atteint une vitesse de défilement convenable. Donc, pour toute écriture, il y a tout d'abord une phase d'accélération, ensuite l'écriture proprement dite, puis une phase de freinage. Les deux phases de mise en vitesse et freinage consomment une certaine longueur de bande. Il n'est pas praticable de consommer cette longueur de bande à chaque écriture d'un nombre. Pour minimiser la place perdue, les données vont être regroupées afin d'exécuter les phases de variation de vitesse pour *tout un groupe de données.*

C'est pourquoi il existe une zone mémoire appelée *tampon* qui regroupe les données avant envoi sur la bande. Sur Amstrad, il y a un tampon de 2K (2048) octets. C'est à ce tampon que correspondent les blocs que le système signale lors d'un LOAD ou d'un SAVE.

Le rôle normal de l'instruction PRINT# est alors simplement de transférer les données dans la zone tampon. Ce n'est que lorsque le 2048e octet du

tampon est écrit que le système d'exploitation décide de lancer le magnétophone et de vider le tampon sur la bande. Ceci est résumé par la figure ci-après.



Comme c'est le fait d'arriver au 2048e octet du tampon qui fait démarrer l'écriture effective sur le magnétophone, il se pose un problème pour le dernier enregistrement. En effet, il y a peu de chance pour qu'un travail donné fasse écrire un nombre d'octets multiple de 2048, et le dernier enregistrement a peu de chances de compter exactement 2048 octets. Par conséquent, il ne devrait pas être écrit.

Eh bien, c'est le rôle de l'instruction CLOSEOUT de s'assurer que tout est bien en ordre pour le fichier avant de l'abandonner, et, s'il reste un enregistrement incomplet non écrit, c'est elle qui le fera écrire.

L'instruction CLOSEOUT est donc fondamentale en écriture. Mais il ne faut pas non plus oublier CLOSEIN en lecture, sinon, vous ne pourrez pas ouvrir d'autres fichiers.

Les retours-chariot

On voit bien que INPUT# et PRINT# jouent respectivement le même rôle que INPUT et PRINT pour le clavier et l'écran. D'ailleurs, INPUT et PRINT sont

les abréviations de INPUT#0 et PRINT#0, c'est-à-dire lecture ou écriture dans la fenêtre 0.

Ceci a pour INPUT#9 une conséquence fondamentale. On sait que l'instruction INPUT au clavier n'accepte jamais plus que l'entrée de 255 caractères consécutifs sans **RETURN** ou **ENTER**. Le dépassement est très grave car il entraîne une perte de contrôle de la machine.

Il en est de même pour INPUT# sur cassette (ou disquette). Par suite, c'est au moment de l'écriture de la cassette que vous devrez *impérativement* vous assurer que vous insérez des caractères retour-chariot de sorte qu'il n'y ait *jamais* plus de 255 caractères entre deux **RETURN/ENTER**.

Comment assurer la présence des retours-chariot ?

Il y a trois méthodes : une implicite qui utilise les propriétés de PRINT/ PRINT#, une méthode explicite et l'utilisation d'une nouvelle instruction : WRITE#.

MÉTHODE IMPLICITE

Vous avez vu que PRINT non terminé par ',' ou ';' fait que le prochain PRINT imprimera à la ligne suivante. C'est parce qu'il y a automatiquement un retour-chariot implicite en fin de ligne dans ce cas. Inversement, ce retour-chariot est supprimé si l'instruction PRINT se termine par ',' ou ';'.

Il en est de même avec PRINT#9. Donc le secret est de faire des ordres de la forme :

PRINT#9, [moins de 255 caractères] [pas de ',' ni ';'].

Cette condition est remplie à coup sûr si l'on ne met qu'une variable dans chaque ordre d'impression, comme nous le conseillons plus haut :

```
PRINT#9,A
PRINT#9,A$
```

Il faut d'ailleurs noter que s'il y a plusieurs variables dans la liste, il faut écrire :

```
PRINT#9,A;",";B
```

afin d'assurer que l'instruction INPUT# associée trouve bien la virgule qui sépare deux données.

Si l'on écrivait PRINT#9,A,B il n'y aurait pas de virgule sur la bande ! Il est bien plus prudent de ne mettre qu'une variable par instruction, ou alors utilisez WRITE# (voir plus loin).

Corollairement, si l'on écrit une chaîne de caractères, il ne faut pas qu'elle contienne de virgule ou de deux-points, sinon l'INPUT# s'arrêtera à cette ','

ou ce ':' comme le ferait INPUT. Ce problème peut se résoudre par l'emploi de l'instruction LINE INPUT/LINE INPUT# ou alors de WRITE#.

MÉTHODE EXPLICITE

Vous avez un meilleur contrôle de ce que vous écrivez sur la bande en n'écrivant que des chaînes de caractères A$ ou STR$(nombre) et en envoyant explicitement le retour-chariot par CHR$(13) (13 est le code ASCII de retour-chariot) :

```
PRINT#9,A$;CHR$(13);
```

Le ';' terminal est nécessaire, sinon il y aurait deux retours-chariot consécutifs, ce qui ne gênerait pas la lecture de A$ mais donnerait la chaîne vide pour la lecture de la variable suivante.

Cette forme d'écriture est fortement conseillée tant pour les disques que les cassettes. Nous l'utiliserons dans l'exemple de la fin du chapitre.

L'INSTRUCTION WRITE

Cette instruction a exactement la même forme que PRINT, par exemple :

```
WRITE A,B$
WRITE#9,A;B,C$
```

La différence est que les nombres sont imprimés sans espace autour, les chaînes de caractères sont imprimées entre guillemets et surtout, que l'on ait employé une ',' ou un ';' pour les séparer, deux éléments sont imprimés séparés par une virgule. Voilà le moyen d'assurer la présence de virgules sur la bande.

Exercice 1.3. – On a un tableau de 10 chaînes de 47 caractères chacune. Transférez-les sur cassette.

La fonction EOF

Lors de l'exercice 1.2., nous avons relu complètement un fichier que nous avions écrit. Comme nous savions que nous avions écrit 50 nombres, un problème nous a été évité : détecter la fin du fichier.

Dans le cas d'un fichier inconnu ou dans certains traitements moins simples, le problème se pose. Il faut être capable de déceler la fin du fichier.

La fonction EOF est là pour ça : elle n'a pas d'argument puisqu'il ne peut y avoir qu'un fichier en jeu. Elle vaut 'vrai' lorsque la fin de fichier a été atteinte. Ainsi, lire un fichier en entier pourrait s'écrire :

```
50 WHILE NOT EOF
60 INPUT#9,...
...
80 WEND
```

Exercice 1.4. – Retraiter l'exercice 1.2. en supposant que le nombre de données à lire n'est pas connu.

EXEMPLE DE MINI-SYSTÈME DE DONNÉES

Il est temps maintenant de voir concrètement ce que les traitements de fichiers permettent de faire.

Pour cela, nous allons constituer une mini-base de données regroupant des informations sur nos amis et relations afin de faciliter l'organisation de réceptions amicales. Outre le fichier qui regroupe les informations, le système comprendra deux programmes : le programme de saisie des informations et création du fichier et le programme de consultation du fichier. Nous avons choisi ici à dessein une application de type informatique personnelle. Pour les autres types d'applications (professionnelles, enseignement), la marche à suivre serait la même.

Toute *base de données* (auparavant l'on disait "banque de données") est constituée de *fichiers* qui regroupent les informations de la base et de *programmes* d'entretien et d'exploitation (le plus souvent par consultation) des fichiers. Ici, ces différents éléments se réduisent à leur plus simple expression : un seul fichier, un programme de consultation.

Un des premiers choix à effectuer pour la constitution d'une base de données est de décider quelles informations celle-ci contiendra. Ce choix doit, bien sûr, se faire en fonction de l'application envisagée.

Dans notre cas, que voulons-nous ? Des renseignements sur les gens que nous connaissons pour nous aider à décider qui nous invitons à notre prochaine réception. De quoi avons-nous besoin pour chaque personne ? De son nom, prénom, adresse (pour envoyer l'invitation), téléphone, puis de renseignements tels que : ami, relation professionnelle ou membre de la famille, ou encore tels que "de compagnie agréable" ou "à n'inviter qu'avec précautions" ou bien "gros mangeur" ou "difficile en cuisine"...

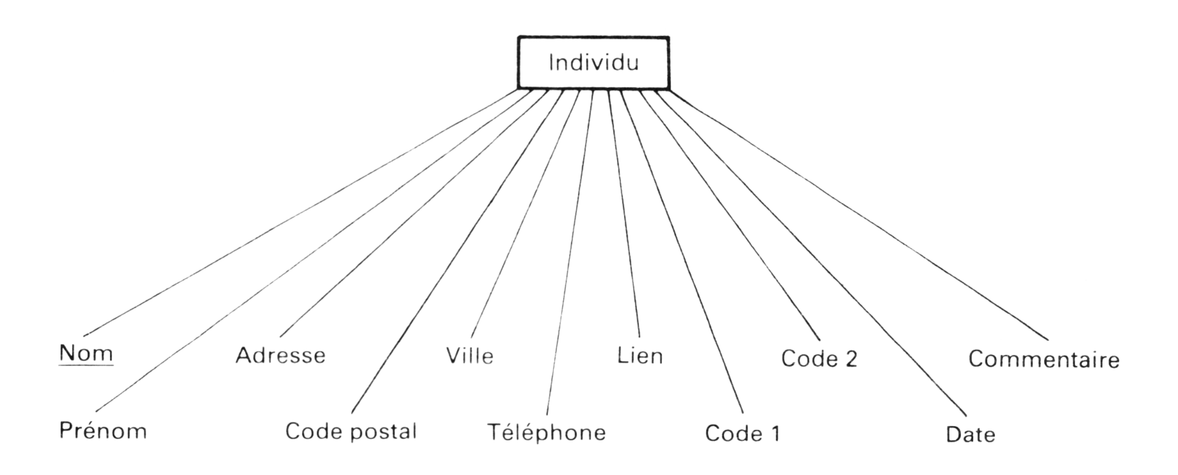
Ces derniers renseignements seront, en principe, stockés sous forme codée, ce qui explique pourquoi les fichiers informatiques sont parfois mystérieux. Mais il suffit de connaître le code (par exemple F=famille, A=ami, R=simple relation) ou d'avoir des programmes qui les restituent en clair.

Toutes ces informations doivent être ensuite organisées sur le fichier. Bien entendu, tous les renseignements qui concernent un même individu seront regroupés.

Ces renseignements constituent un *article* et le fichier est formé de la suite des articles correspondants à chacun des individus présents.

Chacune des informations concernant un individu constitue une *rubrique* à l'intérieur de l'article. Bien sûr, les différentes rubriques sont toujours *dans le même ordre* dans tous les articles et c'est cet ordre qui définit la structure du fichier et son organisation.

Cette organisation peut se représenter par un dessin, le *type-article* que, pour notre application, nous proposerons ainsi :



Nous avons ici 10 rubriques. Ce nombre est adopté dans la plupart des programmes de gestion des informations personnelles commercialisés, mais on pourrait adopter tout autre nombre.

Nom est souligné car il peut servir de clé de classement des articles.

Le type-article peut être représenté avec plus de précision par un tableau comme celui ci-dessous. Le nombre de caractères (ou de chiffres pour du numérique) peut être laissé libre (ce que nous ferons dans un premier temps) ; il peut être limité (pour limiter la taille du fichier) ou même imposé (articles de longueur fixe).

Rubrique	Libellé	Explication	Nombre car.
1	Nom		Libre mais raisonnable
2	Prénom		
3	Adresse		
4	CPVille	Code postal ville *Ex.* – 75010 PARIS	
5	Téléphone		
6	Lien	F=famille A=ami R=relation (*ex.* professionnelle)	1. car.
7	Code 1	Définit l'agrément d'inviter la personne. *Ex.* A=agréable, B=moyen, C=à éviter	1 car.
8	Code 2	Définit le comportement à table. *Ex.* G=gourmand, C=chipoteur, D=difficile	1 car.
9	Date	Permet de se rappeler un anniversaire à souhaiter sous la forme jjmmaa. *Ex.* 200544	6 car.
10	Commentaire	Permet de rappeler une particularité. Va de "RAS" à "NE PAS INVITER EN MEME TEMPS QUE TANTE ELSA"	Libre

Type-article "INDIVIDU" de notre fichier

Bien entendu, vous êtes libre de modifier ces rubriques à votre gré. Nous avons ici choisi toutes les rubriques sous forme de chaîne de caractères : c'est en fait le plus commode. Tous les programmes d'exploitation du fichier devront se référer à ce tableau pour retrouver les informations voulues.

Nous sommes maintenant en mesure d'écrire le premier programme de création du fichier. Ce programme est en fait extrêmement simple : il est formé d'une suite d'interrogations permettant d'entrer les différentes rubriques par INPUT. On remarquera que nous faisons les écritures conformément à la recommandation d'inscrire les retours-chariot explicitement.

La manière dont nous traitons la fin du fichier est, elle aussi, particulière. Pour signifier au programme que nous ne voulons plus entrer d'individus, nous fournissons comme nom "$$$" (en informatique, il y a toujours des dollars à la clé !).

Ce "nom" particulier est écrit sur la bande : il sera reconnu comme fin de fichier par les programmes d'exploitation.

L'instruction 15 permet au programme de fonctionner tant sur cassette que sur disquette, à la demande de l'utilisateur.

```
1   REM CREATION CASSETTE
5   CR$=CHR$(13)
10  INPUT "NOM DU FICHIER, CASS OU DISQ";F$,P$
15  IF P$="C" THEN |TAPE ELSE |DISC
20  OPENOUT F$
30  INPUT "NOM";A$
35  PRINT#9,A$;CR$;
37  IF A$="$$$" GOTO 140
40  INPUT "PRENOMS";A$
45  PRINT#9,A$;CR$;
50  INPUT "ADRESSE";A$
55  PRINT#9,A$;CR$;
60  INPUT "CP VILLE";A$
65  PRINT#9,A$;CR$;
70  INPUT "TELEPHONE";A$
75  PRINT#9,A$;CR$;
80  INPUT "LIEN:F,A OU R";A$
85  PRINT#9,A$;CR$;
90  INPUT "CODE1:A,B OU C";A$
95  PRINT#9,A$;CR$;
100 INPUT "CODE2:G,C OU D";A$
105 PRINT#9,A$;CR$;
110 INPUT "DATE";A$
115 PRINT#9,A$;CR$;
120 INPUT "COMMENTAIRE";A$
125 PRINT#9,A$;CR$;
130 GOTO 30
140 CLOSEOUT : PRINT "FIN CREATION"
```

Exercice 1.5. – Ajoutez un troisième code en une lettre.

Exercice 1.6. – Limiter à 20 caractères les rubriques 1, 2, 3 et 10.

Nous vous conseillons instamment d'implanter ce programme et de constituer un fichier de données réelles. Pas besoin d'y mettre des centaines d'ar-

ticles : une vingtaine suffira pour vous rendre compte de l'intérêt de son utilisation.

Faisons maintenant un programme de consultation de notre mini-base de données. Le principe est que le programme parcourt le fichier d'un bout à l'autre et imprime sur l'écran les articles qui satisfont à certaines conditions à mesure qu'il les rencontre. Par exemple, on peut demander de lister toutes les personnes de la famille, c'est-à-dire tous les articles dont la 6e rubrique est "F", ou tous les invités agréables (7e rubrique = "A"), etc.

Le programme commence par demander sur quel numéro K de rubrique porte la sélection et quelle est la valeur recherchée. Notez que si l'on donne K=0, il n'y a pas de sélection : tous les articles sont listés. Afin de pouvoir examiner les articles les uns après les autres sur l'écran, l'utilisateur doit taper sur une touche pour passer au suivant.

Voici le programme :

```
1   REM UTILISATION CASSETTE
10  INPUT "NOM DU FICHIER, CASS OU DISQ";F$,P$
15  IF P$="C" THEN |TAPE ELSE |DISC
20  OPENIN F$
30  INPUT "NUMERO,VALEUR RECHERCHEE";K,K$
40  DIM A$(10)
50  INPUT#9,A$(1)
60  IF A$(1)="$$$" GOTO 150
70  FOR I=2 TO 10
75  INPUT#9,A$(I) : NEXT
80  IF K=0 GOTO 90
85  IF A$(K)<>K$ GOTO 50
90  CLS : FOR I=1 TO 10
95  PRINT A$(I) : NEXT
100 PRINT:PRINT:PRINT
105 PRINT "TAPER SUR ESPACE POUR CONTINUER
    OU RETURN/ENTER POUR TERMINER"
110 X$=INKEY$ : IF X$="" GOTO 110
120 IF X$=" " GOTO 50
130 IF X$=CHR$(13) GOTO 150
140 GOTO 100
150 PRINT "RECHERCHE TERMINEE" : CLOSEIN
```

Tel qu'il est, ce système de programmes permet de nombreuses applications : quels sont les gens à inviter ? Quel est le numéro de téléphone de Durand ? Etc.

Il est possible de faire mieux.

Exercice 1.7. – Modifier le programme précédent pour que la recherche se fasse sur deux conditions simultanées sur les rubriques n° K et L (donner K = 0 et/ou L = 0 signifie condition indifférente).

Les conditions combinées enrichissent beaucoup le programme. Mais la présentation serait plus commode si, au lieu de donner la rubrique par son numéro, les rubriques étaient citées par leur nom (ou libellé).

Il suffit pour cela de conserver le tableau des libellés dans un enregistrement d'en-tête au début du fichier. Le nombre et le nom des rubriques peuvent alors être définis par l'utilisateur lors de la création du fichier ce qui rend le système applicable dans tous les domaines.

Quant à la présentation des questions, elle se rapproche du langage naturel : beaucoup d'études se font actuellement pour que les bases de données soient consultables en langage naturel (*cf.* fin du chapitre 4).

Une amélioration plus modeste, mais bien utile, serait que la question puisse porter sur une partie de rubrique : par exemple sur le département dans le code postal (quels sont les gens qui habitent dans la Seine-Saint-Denis ?).

Exercice 1.8. – Faites-le. On demande au départ le numéro de rubrique, le numéro du caractère de début et le nombre de caractères concernés.

Une application de ceci : le 15 mai, vous demandez si le 16/5 est une date particulière pour quelqu'un de vos connaissances en donnant comme réponse 9,1,4,1605. Cela peut vous éviter d'oublier l'anniversaire de votre femme (mari) !

Notre système est maintenant assez perfectionné. Son plus grave inconvénient est au niveau de la création : il faut créer le fichier en entier d'un seul coup, et aucune mise à jour n'est possible.

Ces inconvénients se rattachent à deux limitations. La première est due a une caractéristique des cassettes : c'est le fait que l'accès soit séquentiel. La seconde est propre à l'Amstrad : c'est l'impossibilité d'ouvrir plus d'un fichier à un instant donné.

Méthodes d'accès

Les cassettes telles que nous les avons vues souffrent de certaines limitations dues au caractère *séquentiel* du support utilisé. Il est clair qu'une information ne peut être lue sur la cassette que si toutes les informations qui précèdent ont été lues. Couplé avec la lenteur de défilement des cassettes, cela entraîne des attentes fastidieuses.

On dit que l'accès à l'information est *séquentiel* lorsque la seule information accessible est celle devant laquelle le dispositif de lecture est positionné. C'est le cas des cassettes. On dit que l'accès à l'information est *direct* lorsque, quelle que soit la position (résultant de la lecture précédente) dans laquelle se trouve la tête de lecture, il est possible d'accéder presque instantanément à n'importe quelle autre information désignée par son adresse. C'est le cas des disques, mais le BASIC de l'Amstrad ne permet pas d'en tirer parti. Nous donnerons au chapitre 3 un moyen de simuler l'accès direct sur Amstrad.

Bien sûr, l'accès direct permet des traitements plus rapides et plus efficaces. Mais les limitations des cassettes ne sont pas seulement dues à l'accès séquentiel. Elles sont dues à l'impossibilité pour les cassettes de passer, sans l'intervention de l'opérateur, de la lecture à l'écriture, ou du réembobinage à la lecture, ou de l'avance rapide à la lecture.

Si ces manœuvres étaient possibles, on pourrait, sous certaines conditions, comme cela peut se faire sur les unités de bande des gros ordinateurs, simuler en partie l'accès direct. L'emploi dans les micro-ordinateurs de magnétophones style grand public nous l'interdit.

Une autre limitation qui joue sur les unités de bande même les plus chères, est l'impossibilité de réécrire sur place une donnée, donc l'impossibilité de mise à jour. Ceci est dû à la position de la tête d'effacement *en avant* de la tête de lecture/écriture. Cet inconvénient n'existe pas avec les disques, où il y a une seule tête qui fait tout. C'est de toutes façons impossible sur Amstrad puisqu'on ne peut pas ouvrir un fichier en lecture/écriture.

Une autre limitation qui, elle non plus, n'est pas due uniquement à l'accès séquentiel est que, à un instant donné, on ne peut avoir sur une cassette qu'un fichier actif (c'est-à-dire ouvert). En effet, la cassette ne peut être positionnée qu'à un endroit à la fois, et il n'est pas possible de changer d'endroit puisque l'ordinateur ne peut pas sans intervention de l'opérateur mettre en réembobinage ou en avance rapide. Cette limitation-là est peut-être la plus gênante en gestion (à part la lenteur) : en effet, il est très souvent nécessaire d'accéder simultanément à plusieurs fichiers même si chacun d'eux est parcouru en séquence.

Cette limitation n'a pas de raison physique d'exister sur disque. Malheureusement, elle est effective sur Amstrad puisque le BASIC de l'Amstrad interdit d'ouvrir plusieurs fichiers simultanément. Ceci interdit des opérations telles que la mise à jour d'un fichier. On ouvrirait en lecture le fichier à mettre à jour et en écriture le fichier destiné à contenir la nouvelle version. Les informations seront lues sur le premier fichier et recopiées telles quelles sur le second tant qu'aucune modification ne devra être apportée à un enregistrement. Lorsqu'une modification devra être apportée, l'information sera lue sur le premier et c'est l'information modifiée qui sera écrite sur le second fichier. Nous simulerons ces possibilités au chapitre 3 en faisant intervenir la mémoire.

Nous allons maintenant étendre notre champ d'action en dotant notre système de la possibilité d'impression durable de résultats.

CHAPITRE 2
L'IMPRIMANTE

GÉNÉRALITÉS

L'écran est un excellent moyen de restitution des résultats de l'ordinateur. Il est en particulier rapide et silencieux. Mais il a un inconvénient grave : il ne délivre pas un document durable ; cela se manifeste dès que l'information à restituer dépasse une page d'écran. Or, certaines applications nécessitent impérativement un document durable : listing d'un programme à conserver, bilan, bulletin de paie, etc.

Il existe des unités, analogues aux photocopieurs, qui savent reproduire sur papier par analyse du signal vidéo le contenu d'un écran (on les appelle unités de "hard-copy"). Mais leur prix les rend, pour le moment, inaccessibles à l'informatique individuelle.

La solution est donc apportée par les imprimantes. Toute une gamme d'imprimantes simplifiées a été développée (et est en cours de développement) pour l'informatique individuelle. Elles ne ressemblent pas aux imprimantes de gros ordinateurs mais plutôt aux imprimantes des consoles-opérateur ou des terminaux. Ces dernières années, leur rapport performances-prix a fait des progrès étonnants.

CONNEXION A L'AMSTRAD

Il y a un très grand nombre d'imprimantes connectables à l'Amstrad (de ce point de vue, les trois modèles 464, 664 et 6128 ont exactement le même comportement).

En effet, outre les deux modèles de la marque Amstrad, la DMP 1 et la DMP 2000, on peut connecter presque toutes les imprimantes du marché de la micro-informatique. En effet, l'interface imprimante de l'Amstrad se conforme à l'interface standard du marché appelé "interface Centronics".

Toutefois, l'Amstrad s'éloigne de la norme Centronics sur deux points :

– le connecteur n'a pas la forme normalisée. Il vous faudra donc vous procurer un câble adéquat. Il y en a un schéma dans le n° 1 de la revue *MICROSTRAD*, si vous voulez le faire vous-mêmes ;

– parmi les signaux transmis, il y a normalement 8 bits de données utilisés pour fournir à l'imprimante le code du caractère à imprimer. L'Amstrad ne fournit que 7 bits, le bit le plus significatif étant toujours à 0. Tant qu'il s'agit de caractères ASCII standards, il n'y a pas de problème : leur code tient en 7 bits. Il y a quelques caractères dont le code est >128, notamment l'accent circonflexe. L'Amstrad effectue automatiquement un transcodage de ces caractères. Mais il n'y a qu'avec une imprimante Amstrad qu'on est sûr d'obtenir les bons caractères. Un autre problème se pose avec les imprimantes à aiguilles en mode graphique. Avec une imprimante à 7 aiguilles, on n'a que des motifs binaires de 7 bits, donc pas de problème ; en revanche, une imprimante qui nécessiterait des motifs binaires de 8 bits serait inutilisable en mode graphique avec l'Amstrad.

QUELLE IMPRIMANTE CHOISIR ?

Gardant ce problème en mémoire, nous passons en revue un certain nombre de paramètres concernant les imprimantes, avec leur impact sur les différentes applications. Ceci devrait vous permettre de circonscrire votre choix en fonction de votre application.

Mode d'impression

C'est le premier paramètre, qui, d'ailleurs, en influence d'autres et notamment le prix.

On distingue les imprimantes :

– **A impact de caractères,** où les caractères gravés en relief sont frappés sur un ruban encré pour s'inscrire sur le papier. C'est le cas des imprimantes à boule ou à marguerite.

– **A aiguilles,** où les caractères sont formés par des aiguilles qui marquent des points dans une matrice 5×7 ou 7×9 (*cf. figure*). La majorité des imprimantes pour micro-ordinateur sont de ce type. Les modèles les plus élaborés ont un mode "qualité courrier" où l'on essaie de diminuer le caractère pointillé du tracé.

```
.oo..
o..o.
o..o.
oooo.
o..o.
o..o.
o..o.
```

Le A en matrice 5×7

– **Thermiques ou électrostatiques.** Les caractères sont à matrice de points comme dans le cas précédent, mais les points sont noircis soit par chauffage, soit par décharges électriques. On peut y rattacher les imprimantes à laser, qui impriment page par page comme une photocopieuse, donc sont très rapides, mais coûtent une fortune.

– **A jet d'encre.** Les points sont formés par l'envoi de gouttelettes d'encre.

Le type d'impression joue sur les trois éléments suivants.

Type de caractères – qualité d'impression

Les imprimantes à caractères tout faits donnent une bien meilleure qualité d'impression que les imprimantes à aiguilles, encore que certaines "qualité courrier" soient très bonnes. En revanche, les imprimantes à matrice ont une plus grande souplesse sur le jeu de caractères : il suffit de changer une MEM pour en changer ; il est même possible d'avoir des caractères programmables.

Type de papier

Les imprimantes à impact (caractères ou aiguilles) admettent, bien sûr, le papier ordinaire. Les imprimantes thermiques ou électrostatiques exigent un papier spécial, souvent cher, et désagréable au toucher. Il existe maintenant des imprimantes à transfert thermique qui utilisent du papier ordinaire.

Vitesse/bruit

Les imprimantes à impact-caractères sont plus lentes que les imprimantes à aiguilles (30 caractères/s. contre 50 à 90 caractères/s.).

Les imprimantes thermiques, électrostatiques ou à jet d'encre sont plus silencieuses que les imprimantes à impact.

Le type de papier est lié à un paramètre important développé ci-après.

Le type d'entraînement du papier

Il existe deux types d'entraînement : *à friction* comme dans les machines à écrire classiques, ou *à picots*. Dans ce dernier cas, le papier se présente sous forme d'une bande continue, pliée en accordéon, percée de trous sur les côtés : les picots entrent dans ces trous pour entraîner le papier.

L'entraînement à friction permet l'utilisation de feuilles volantes de papier ordinaire ou de papier à en-tête, ce qui peut être utile dans certains cas. Il faut cependant vérifier que le mécanisme permette un positionnement initial facile lors de l'introduction du document.

L'entraînement à picots ne laisse place à aucun glissement. Il est seul à permettre la précision de positionnement qu'exigent certaines applications (impression de chèques ou d'étiquettes). Il est alors utile que les picots soient réglables pour autoriser différentes largeurs de papier.

L'idéal est de disposer à la fois des deux sortes d'entraînement. Les imprimantes qui le permettent sont de plus en plus nombreuses, même parmi les imprimantes bon marché.

Enfin, intervient le paramètre essentiel suivant :

La largeur d'impression ou nombre de caractères par ligne

Il faut au moins 40 caractères par ligne. La plupart des imprimantes possèdent 80 colonnes (vieille survivance des terminaux ; c'est le nombre de colonnes d'une carte perforée classique), ce qui est parfait pour la plupart des applications.

D'autres largeurs existent : 120 ou 132 colonnes.

Les deux imprimantes de marque Amstrad, la DMP 1 et la DMP 2000, ont 80 colonnes par ligne. Nous étudions leur utilisation maintenant.

UTILISATION DE L'IMPRIMANTE AMSTRAD DMP 1

La DMP 1 est une imprimante à aiguilles, 80 colonnes, entraînement à picots.

Pour son utilisation en BASIC, il n'y a presque rien à dire : les éléments nous sont déjà parfaitement connus. Il suffit de savoir que tout se passe comme si l'on écrivait dans la fenêtre d'écran n° 8.

Exemple

Pour imprimer 'Bonjour', il suffit de :

```
PRINT#8, "Bonjour"
```

Toutes les propriétés de l'instruction PRINT sur écran se retrouvent. Notez que, comme pour les fenêtres d'écran et à la différence des cassettes ou disquettes, il n'y a pas besoin d'OPEN ni de CLOSE.

Cette analogie entre impressions sur imprimante et sur écran permet d'envoyer à volonté les impressions sur écran ou imprimante. Par exemple, pendant les essais d'un programme, on imprimera sur écran pour éviter de gâcher du papier. Une autre application consiste à laisser le choix du périphérique de sortie au début de l'exécution du programme.

Exercice 2.1. – Faites-le : le programme demande à l'utilisateur s'il veut imprimer sur écran ou imprimante.

Exercice 2.2. – Puisque vous savez imprimer, imprimez une table des carrés des nombres de 1 à 100 sur deux colonnes avec un titre.

PROBLÈMES D'ALIGNEMENT – IMPRESSION FORMATÉE

Le programme de l'exercice précédent produit des impressions satisfaisantes, sauf sur un point : les nombres sont alignés sur la gauche et non sur la droite (on note que les ',' et ';' agissent comme dans PRINT). Pour obtenir

l'alignement sur la droite, de même que pour imprimer un nombre raisonnable de décimales (et les points alignés), il faut faire appel à une fonction de formatage.

On peut pour cela combiner des fonctions chaînes de caractères. Mais le BASIC de l'Amstrad a beaucoup mieux : la clause USING.

PRINT USING

Utilisable tant sur écran avec PRINT simple que dans une fenêtre ou sur imprimante avec PRINT#, la clause USING permet de spécifier un *modèle* (nombre total de caractères utilisés, nombre de chiffres après et avant le point, etc.) pour chaque élément à imprimer.

Les instructions sont de la forme :

PRINT [#n,] [liste1;] USING format ; liste2 [;]

où n est le n° de fenêtre ou le 8 désignant l'imprimante (les crochets montrent que cet élément est facultatif), liste1 est une liste d'éléments éventuels à imprimer sans format, liste2 est une liste d'éléments à imprimer suivant le format 'format'. Le format est une chaîne de caractères spécifiée soit entre guillemets, soit sous forme d'une variable alphanumérique.

Dans liste2, tous les éléments doivent être du même genre car ils doivent être imprimables avec le même format. Il ne peut y avoir qu'une clause USING dans une instruction PRINT. Le séparateur entre éléments dans liste2 peut être ',' ou ';' mais cela ne fait pas de différence : on obtient toujours l'effet de ';'. Le format doit être séparé de liste2 par un ';'.

Le principe fondamental du format est que celui-ci est une *image* de la sortie qu'on veut obtenir. Chaque caractère du format correspond en principe à un caractère de la sortie, par exemple, PRINT USING "##" ; A fait sortir le nombre A sur deux chiffres puisqu'il y a deux signes dièse.

FORMATS NUMÉRIQUES

Le signe le plus fondamental dans un format numérique est le # : il représente un chiffre. Si le nombre à sortir a moins de chiffres, il est imprimé cadré à droite dans la zone de largeur égale au nombre de dièses et le système met des blancs à gauche.

Exemple

Format	Nombre	Sortie
###	5	5
###	24	24
###	112	112
###	6538	%6538

Le dernier exemple montre que si le format est trop petit (à cause du nombre de chiffres ou s'il faut une place pour le signe −), le nombre est tout de même imprimé complet, mais l'anomalie est signalée par un '%'.

On peut inclure parmi les # un point (il ne doit y en avoir qu'un) qui marque la place du point décimal. Par exemple ###.## signifie qu'on veut 3 chiffres pour la partie entière et 2 chiffres pour la partie décimale. Ceci résout le problème de sortir toujours le même nombre de décimales et d'aligner les points.

Le format peut être précédé du signe +, ce qui signifie qu'on veut que le signe soit imprimé même si le nombre est positif. Le signe + peut être à la fin du format, ce qui signifie qu'on veut que le signe suive le nombre imprimé (cela sert en comptabilité). On peut mettre un signe − en fin de format : cela signifie que le signe ne doit apparaître que si le nombre est négatif, et qu'il doit suivre le nombre.

Le format peut se terminer par quatre flèches vers le haut. Cela signifie qu'on veut le nombre sous forme exponentielle :

PRINT USING "##.## ^^^^" ;

1.456 fait imprimer 1.46E+00 (remarquer aussi l'arrondi de la partie décimale).

Le format peut commencer par ** : à ce moment, au lieu de compléter par des blancs à gauche, on imprimera des * ; cela sert pour écrire la somme sur un chèque.

Le format peut commencer par $$ ou ££ : à ce moment, le nombre sortira précédé de $ ou £, ce qui est utile pour un prix.

Tout chaîne de caractères autres que ceux que nous avons cités peut être mise au début et/ou à la fin d'un format numérique : elle sera purement et simplement recopiée pour accompagner le nombre.

FORMATS ALPHANUMÉRIQUES

Ils sont moins intéressants que les numériques, mais ils permettent cadrages et troncatures.

PRINT USING "\   \" ; A$ (il y a 5 espaces) fait imprimer A$ suivie d'espaces pour compléter à 7 caractères, si A$ fait moins de 7 caractères. Si A$

fait plus, les 7 premiers caractères de A$ sont imprimés. Vous voyez que les \ comptent.

Le format "!" fait imprimer le premier caractère de la chaîne à imprimer.

Le format "&" a une action plus subtile. En l'absence de format, si, au moment d'imprimer une chaîne, le système s'aperçoit qu'il ne reste pas assez de place sur la ligne, il va à la ligne avant d'imprimer. Avec "&", cet effet est supprimé et la chaîne se trouvera à cheval sur deux lignes.

Exercice 2.3. – Dressez la même table des carrés qu'à l'exercice précédent, mais alignez les nombres sur la droite.

Exercice 2.4. – Dressez une table de la fonction sin x entre O et PI/4 par pas de 0,01. Mettre un titre à la table et la présenter sur 5 doubles colonnes :

x	sin x	x	sin x	x	sin x	x	sin x	x	sin x

L'instruction WIDTH

Cette instruction sert à décider de la largeur des impressions sur imprimante. Elle est de la forme :

WIDTH n

où n doit être compris entre 1 et 255.

Si la largeur spécifiée est supérieure à la largeur physique de l'imprimante, l'impression continue à la ligne. Si la largeur spécifiée est plus petite, le système ajoute les passages à la ligne nécessaires pour que tout se passe comme si l'imprimante avait la largeur indiquée.

Ceci est utile surtout pour limiter la largeur des listings. Au fait, comment fait-on un listing sur imprimante ?

Obtention d'un listing

Il est intéressant de pouvoir faire le listing d'un programme sur imprimante afin de l'examiner à loisir ou de le conserver.

Il suffit de compléter le LIST par un #8. Attention, si vous spécifiez un intervalle de lignes à lister, le #8 va après, par exemple :

```
LIST #8      (listing du programme entier)
LIST 100-10000,#8
```

CARACTÈRES DE CONTRÔLE

Etant un périphérique "intelligent", la DMP 1 reconnaît certains caractères ou certaines séquences de caractères pour produire certains effets spéciaux.

On envoie ces caractères par :

PRINT#8, CHR$ (code correspondant) ;

Voici le tableau des codes (ils sont en décimal – lorsqu'il y a deux codes, on les envoie successivement) :

Code	Fonction
10	Passage à la ligne.
13	Retour-chariot (avec passage à la ligne).
20	Retour-chariot (sans passage à la ligne).
14	Passage en caractères élargis.
15	Retour aux caractères normaux ou sortie du mode graphique.
16	Positionnement en unités caractères.
27 16	Positionnement en unités points élémentaires.
27 75	Passage en mode graphique.
28	Répétition de motif en mode graphique.

Nous donnons maintenant quelques explications utiles. Mais le mieux est que vous fassiez des essais si vous possédez une imprimante.

Les retours-chariot

L'imprimante fonctionne, elle aussi, avec une mémoire tampon, c'est-à-dire qu'elle n'imprime pas aussitôt les caractères qu'on lui envoie. Elle les accumule dans un tampon de 80 caractères et elle les imprime, soit lorsque le tampon est plein, soit lorsqu'elle reçoit un caractère de code 10, 13 ou 20.

Dans ce dernier cas, elle imprime le contenu du tampon, puis fait avancer le papier, de façon différente selon le code.

Avec le code 10, il y a passage à la ligne en restant à la même colonne que précédemment.

Avec le code 20, il y a retour-chariot pur, c'est-à-dire qu'on revient en début de la ligne où l'on était déjà. C'est utile pour faire des impressions superposées.

Avec le code 13, on a la combinaison des deux autres codes, c'est-à-dire un retour en début de ligne suivante. Il faut noter que sur DMP 1, il y a un interrupteur qui détermine le comportement du code 13 : vous pouvez le rendre identique au code 20. Notez aussi que vous n'avez en principe pas à vous préoccuper du retour-chariot : un

```
PRINT#8, "Bonjour"
```

sans ; terminal est automatiquement suivi d'un code 13.

Impression élargie

Le code 14 fait passer en mode élargi, dans lequel les caractères sont imprimés en matrice 10×7, chaque colonne élémentaire étant redoublée. C'est un bon moyen de mettre une impression en évidence. On revient aux caractères normaux à l'aide d'un code 15.

Exemple

```
PRINT#8,"NORMAL";CHR$(14);"ELARGI";CHR$(15);"NORMAL"
```

produit l'impression

NORMAL**ELARGI**NORMAL

Positionnement

A part la fonction TAB qui agit aussi sur écran, l'imprimante DMP 1 a une commande spéciale de positionnement.

Après PRINT#8,A$;B$;C$; où A$=CHR$(16), B$ est le caractère représentant le premier chiffre décimal et C$ le second chiffre du numéro de colonne où l'on veut se positionner, on imprimera en colonnes 10*VAL(B$)+VAL(C$).

Exemple

Si l'on veut imprimer Bonjour en colonne 30, on peut écrire :

```
PRINT#8,CHR$(16);CHR$(51);CHR$(48);"Bonjour"
```

ou

```
PRINT#8,CHR$(16);"30Bonjour"
```

(51 est le code ASCII de 3, 48 celui de 0).

Pour être effectif, un positionnement-caractère doit toujours être suivi de l'impression d'au moins un caractère normal.

On peut faire plusieurs positionnements sur la même ligne, mais ils doivent toujours aller de gauche à droite.

Après un CHR$(16), l'imprimante interprète toujours les deux premiers caractères reçus comme définissant le positionnement. La numérotation se fait à partir de 0 ; dans l'exemple ci-dessus, on imprimera en fait dans la 31e colonne.

POSITIONNEMENT EN COLONNE ÉLÉMENTAIRE

Mais le positionnement peut même se faire à la colonne élémentaire près avec la séquence de contrôle :

```
PRINT#8,CHR$(27);CHR$(16);CHR$(N1);CHR$(N2);
```

Si N est le numéro de colonne élémentaire (de 0 à 479) où l'on veut se positionner, N1 forme les 2 bits les plus significatifs de N et N2 forme les 7 bits de droite de N ; ils se calculent comme suit :

```
N1=INT(N/128)
N2=N AND 127
```

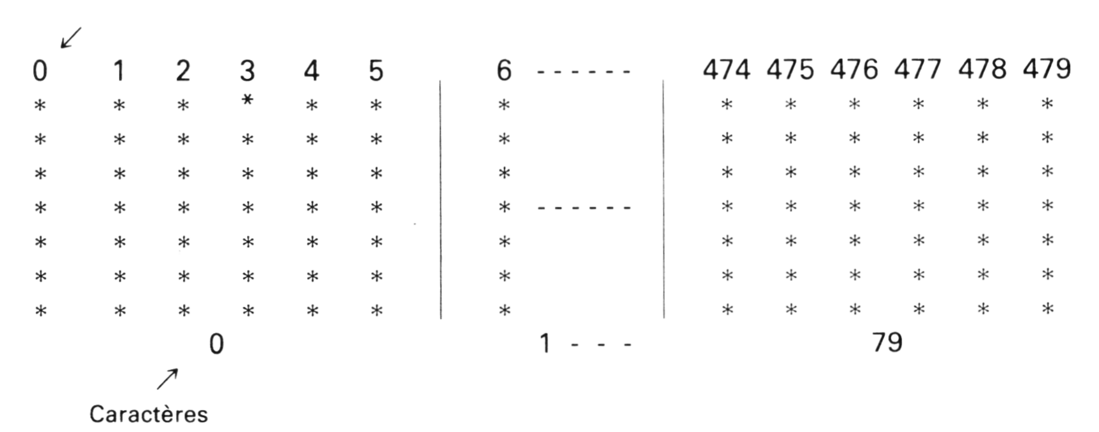


Schéma des deux techniques de positionnement

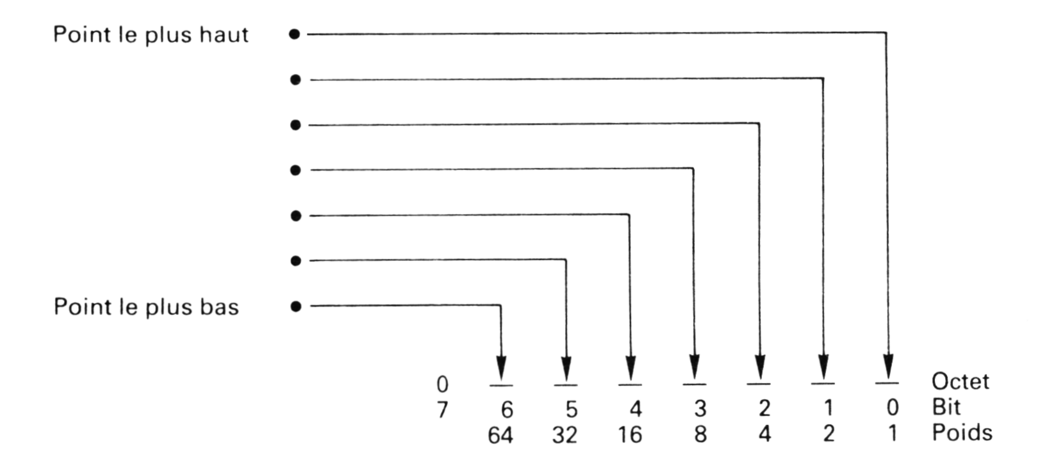
Le mode graphique

On peut spécifier le motif à dessiner dans une colonne élémentaire point élémentaire par point élémentaire. Cela permet soit un dessin pur, soit d'introduire des caractères que l'imprimante n'a pas.

On passe en mode graphique par la séquence :

PRINT#8,CHR$(27);CHR$(75);CHR$(N1);CHR$(N2);

où N1 et N2 sont respectivement les 2 bits les plus significatifs et les 7 bits les moins significatifs du nombre N de colonnes élémentaires qu'on souhaite définir. N est compris entre 1 et 480 et N1, N2 se calculent comme ci-dessus. La séquence doit être suivie de l'envoi de N octets qui vont définir les dessins des N colonnes élémentaires, suivant la correspondance.



Pour passer du dessin à l'octet correspondant, il suffit d'ajouter les poids correspondant aux points qu'on veut noircir. Par exemple, pour le dessin :

```
*
*
o
*
o
*
o
```

on a le motif 1+2+8+32 = 43.

Exercice 2.5. – Imprimez une lettre grecque α.

Lorsque c'est un caractère que vous voulez redéfinir, il faut fournir des données pour 6 colonnes élémentaires.

RÉPÉTITION EN MODE GRAPHIQUE

Lorsque le même motif est à fournir plusieurs fois de suite, on peut utiliser la répétition ; on envoie la séquence :

```
PRINT#8,CHR$(28);CHR$(N);CHR$(M);
```

où N est le nombre de répétitions (compris entre 0 et 127 ; pour signifier 128, on met 0) et M est le motif à répéter.

Exemple

```
PRINT#8,CHR$(28);CHR$(127);CHR$(127)
```

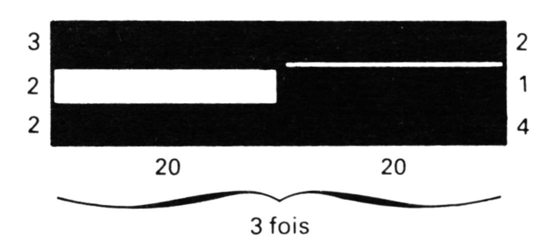
fait imprimer une bande noire.

N.B. – De tels tracés ne sont guère recommandés pour la longévité de l'imprimante.

POUR QUITTER LE MODE GRAPHIQUE

On utilise PRINT#8,CHR$(15).

Exercice 2.6. – Tracez le motif :



Une application évidente de ces barres est le tracé d'histogrammes. C'est l'objet de l'exercice suivant.

Exercice 2.7. – La DATA 1000 donne les effectifs de 10 classes. Tracez l'histogramme correspondant.

```
1000 DATA 120,217,40,75,5,135,180,195,110,30
```

Exercice 2.8. – Réécrire le programme de l'exercice 2.7. mais en écrivant l'effectif en tête de ligne. Les bâtons doivent quand même être alignés sur la gauche.

Caractères soulignés

La DMP 1 n'a pas de mode "caractères soulignés", mais on peut tracer un trait à la ligne suivante sous les mots qu'on veut souligner. On peut faire le trait en mode graphique pour qu'il soit en haut de sa ligne, donc plus près des mots à souligner.

Rappelons qu'il y a d'autres moyens de mettre des mots en évidence : on peut les imprimer en mode élargi ou les taper deux fois en superposition.

Exercice 2.9. – Écrire de trois manières "J'<u>insiste</u> sur ce point" avec insiste souligné.

Obtention des caractères accentués du français

L'imprimante DMP 1 possède les caractères accentués du français. Pour les utiliser, il faut mettre l'imprimante en mode français à l'aide d'interrupteurs situés à l'arrière. A ce moment, vous avez les caractères français à la place d'autres suivant le tableau ci-dessous.

Caractère	Caractère remplacé	Code
à	@	64
ç	\	92
é	{	123
ù	¦	124
è	}	125
^	^	94
tréma	~ (CTRL 2)	126

Caractères français

Pour obtenir une lettre avec accent circonflexe ou tréma, vous devez procéder par superposition. Si vous voulez voir vos lettres accentuées sur l'écran il faut redéfinir le dessin des caractères remplacés à l'aide de SYMBOL.

Exemple

Faites

```
SYMBOL AFTER 64
SYMBOL 64,64,48,120,12,124,140,118,0
```

et vous aurez des 'à' à la place de @ tant sur écran que sur imprimante. (Pour le mode d'emploi de SYMBOL, voyez *La découverte de l'Amstrad*, p. 112.) Nous vous conseillons de faire de même pour les autres caractères accentués.

Réglage de l'interligne

La DMP 1 n'a pas de commande spéciale pour déterminer l'interligne, et pourtant, elle est capable de faire soit du 6 lignes au pouce, soit du 9 lignes au pouce.

Le choix se fait très simplement et la méthode suivie est très judicieuse. L'interligne (1/6e ou 1/9e de pouce) qui sera fait en réponse à un retour-chariot est décidé par le mode (caractères normaux ou graphique) où se trouve l'imprimante au moment où elle reçoit le retour-chariot.

Si l'on est en mode graphique, on fait du 9 lignes au pouce, ce qui rend les lignes jointives : c'est bien adapté au graphique. Si l'on est en caractères normaux, on fait du 6 lignes au pouce, ce qui donne des textes plus lisibles.

Maintenant, si, en mode texte, l'on a incorporé un peu de graphique (ce qui est parfaitement licite) et que l'on doive faire un retour-chariot, on aura un interligne étroit. Si l'on veut garder l'interligne large qui correspond mieux aux textes, il faut, avant le retour-chariot, repasser en mode caractères normaux, par un CHR$(15).

L'IMPRIMANTE TRACEUR DE COURBES

Nous en venons à l'application la plus spectaculaire du mode graphique. Le fait que, en mode graphique, l'interligne soit de 9 lignes au pouce, donc que les lignes soient jointives, permet d'accéder à n'importe quel point élémentaire sur le papier, d'où un tracé de courbes avec une excellente résolution.

La difficulté vient de ce que, pour une macroligne, on a 7 lignes élémentaires qui porteront chacune au moins un point de la courbe. Ces points devront être tracés de gauche à droite, donc il faudra les classer au préalable.

Le mieux est de préparer le dessin de la macroligne dans un tableau de 80 nombres (un pour chaque colonne élémentaire) M(K). Chaque nombre est initialisé à 0. Ensuite, pour noircir le point colonne élémentaire K (de 0 à 479), ligne élémentaire L (de 0 à 6), on fait M(K)=M(K) OR 2^L.

Une fois la ligne prête, il n'y a plus qu'à l'imprimer par :

```
FOR K=0 TO 479 : PRINT#8,CHR$(M(K)); : NEXT
```

A titre d'exemple, nous traçons une période de sinusoïde :

```
10 DIM M(479)
20 DEF FNF(X)=INT(240+200*SIN(PI*X/70))
30 FOR X=0 TO 139 STEP 7
40 FOR K=0 TO 479 : M(K)=0 : NEXT
50 FOR L=0 TO 6 : K=FNF(X+L)
60 M(K)=M(K) OR 2^L
70 NEXT
80 PRINT#8,CHR$(27);CHR$(75);CHR$(3);CHR$(95);
90 FOR K=0 TO 478 : PRINT#8,CHR$(M(K)); : NEXT
100 PRINT#8 : NEXT
```



Pas mauvais résultat pour une imprimante de cette classe de prix, n'est-ce-pas ?

Mais, bien sûr, le tracé est épouvantablement long. Il pourrait être accéléré en évitant d'imprimer les blancs et, surtout, en faisant appel au langage machine.

Exercice 2.10. (difficile) – Ayant un dessin haute résolution sur l'écran, le copier sur l'imprimante.

N.B. – Pour utiliser convenablement le mode graphique, il faut faire WIDTH 255, sinon, le système insère des retours-chariot intempestifs qui apparaissent comme des dessins imprévus.

UTILISATION DE L'IMPRIMANTE DMP 2000

Ce nouveau modèle est destiné à remplacer la DMP 1 ; de prix voisin, il offre beaucoup plus de possibilités. Citons entre autres la double possibilité d'entraînement, la qualité courrier, l'italique, plusieurs largeurs de caractères, exposants et indices, l'espacement proportionnel et plusieurs densités graphiques. On le voit, cela fait beaucoup de caractéristiques favorables pour l'application "traitement de textes".

Les interrupteurs de mode

Le fonctionnement de l'imprimante obéit à certaines options décidées par 16 petits interrupteurs situés à l'arrière à côté de la prise Centronics de liaison avec l'ordinateur. De toutes façons, ces options peuvent toujours être décidées par logiciel (grâce à l'envoi de codes de contrôle) ; il y a donc un peu double emploi.

Toutefois, il peut être utile de fixer par les interrupteurs des options que vous êtes sûrs de vouloir toujours. Dans ce contexte, les plus intéressantes sont le jeu de caractères national (inter 1–1 à 1–3) et la qualité courrier (inter 1–8).

Si vous voulez surtout faire des listings de programmes, activez le jeu de caractères américain (inter 1–1 à 1–3 dans les positions ON ON ON) pour avoir les caractères @ \ ¦ etc.

Au contraire, si vous voulez les lettres accentuées du français, mettez les interrupteurs dans les positions OFF ON ON respectivement. Les caractères remplacés sont les mêmes que sur la DMP 1.

Si vous voulez toujours être en qualité courrier, mettez l'inter 1–8 en position ON. Attention, cela vous interdit certaines possibilités comme gras ou double frappe (sauf si vous revenez à la qualité ordinaire par logiciel).

Les interrupteurs doivent être manipulés avec l'imprimante hors tension ; de toutes façons, il faudrait l'éteindre puis la rallumer pour que le nouvel état des interrupteurs soit pris en compte (ils sont "lus" par la routine de RESET de l'imprimante).

Caractères de contrôle

La DMP 2000 reconnaît beaucoup trop de combinaisons de caractères de contrôle pour que nous puissions toutes les étudier en détail. Il y a 4 catégories de commandes :

1. – commandes d'aspect d'impression ;
2. – commandes de paramétrage d'impression ;
3. – commandes graphiques ;
4. – commandes diverses.

Dans la suite, nous définirons les séquences sous forme de suites de nombres (en décimal). Pour envoyer la séquence, on fait :

```
PRINT#8,CHR$(nb1);CHR$(nb2)...
```

COMMANDES D'ASPECT D'IMPRESSION

L'aspect d'impression peut être varié suivant quatre dimensions (ou directions) :

– **le style** : standard, mini, condensé, proportionnel (l'écart entre caractères est variable), qualité courrier et proportionnel qualité courrier ;

– **le type** : normal, gras, italique et italique gras ;

– **l'apparence** : normal, double frappe, indice et exposant ;

– **l'emphase** : normal, double largeur, souligné et souligné double largeur.

Cela fait théoriquement 216 combinaisons possibles !! Mais toutes ne sont pas permises et certaines font double emploi. Le tableau suivant résume les combinaisons permises, sachant que les 4 options d'emphase sont applicables à toutes. Le tableau donne aussi la suite des codes de contrôle à envoyer pour activer l'option et la désactiver.

STYLES	TYPES			
	Normal	**Gras 27,69 27,70**	**Italique 27,52 27,53**	**Italique gras 27,52,27,69 27,70,27,53**
Standard	Normal Double frap. – Indice – Exposant –	Normal Double frap. – Indice – Exposant –	Normal Double frap. 27,71/27,72 Indice 27,83,1/27,84 Exposant 27,83,0/27,84	Normal Double frap. – Indice – Exposant –
Mini 27,77 27,80	Normal Double frap. Indice Exposant		Normal Double frap. Indice Exposant	
Propor-tionnel 27,112,1 27,112,0	Normal Double frap.		Normal Double frap.	
Condensé 15/18	Normal Double frap. Indice Exposant		Normal Double frap. Indice Exposant	
Courrier 27,120,1 27,120,0	Normal Indice Exposant			
Courrier prop.+	Normal			

Aspects d'impression

Notes

– Les séquences de contrôle n'ont été citées qu'une fois.
\+ Pour passer en courrier proportionnel : 27,120,1,27,112,1.
Pour le supprimer : 27,112,0,27,120,0.

Le programme ci-dessous fait imprimer toutes les combinaisons possibles. xxx$ est le nom d'un mode d'impression, xxxa$ est la chaîne qui active ce mode et xxxs$ est la chaîne qui le supprime.

```
10 FOR emp=1 TO 3 : READ emp$(emp)
20 GOSUB 1000 : empa$(emp)=a$
30 GOSUB 1000 : emps$(emp)=a$
40 NEXT
50 FOR sty=1 TO 6 : READ sty$(sty)
70 GOSUB 1000 : stya$(sty)=a$
80 GOSUB 1000 : stys$(sty)=a$
90 NEXT
100 FOR typ=1 TO 4 : READ typ$(typ)
110 GOSUB 1000 : typa$(typ)=a$
120 GOSUB 1000 : typs$(typ)=a$
130 NEXT
140 FOR app=1 TO 4 : READ app$(app)
150 GOSUB 1000 : appa$(app)=a$
160 GOSUB 1000 : apps$(app)=a$
170 NEXT
200 FOR emp=1 TO 3
205 PRINT#8,empa$(emp);emp$(emp)
210 FOR sty=1 TO 6
220 FOR typ=1 TO 4
230 PRINT#8,stya$(sty);typa$(typ);sty$(sty);typ$
(typ);
240 FOR app=1 TO 4
250 PRINT#8,appa$(app);app$(app);apps$(app);
260 NEXT : PRINT#8
270 PRINT#8,typs$(typ);stys$(sty);
280 NEXT:NEXT
285 PRINT#8,emps$(emp) :NEXT
290 END
1000 REM lecture de sequence de controle
1010 REM
1020 a$="":READ a :WHILE a<>-1
1030 a$=a$+CHR$(a):READ a : WEND
1040 RETURN
5000 REM les data
5010 DATA Simple,-1,-1
5020 DATA Double largeur,27,87,1,-1,27,87,0,-1
5030 DATA Souligne,27,45,1,-1,27,45,0,-1
5050 DATA Standard,-1,-1
5055 DATA Mini,27,77,-1,27,80,-1
5060 DATA Proportionnel,27,112,1,-1,27,112,0,-1
5065 DATA Condense,15,-1,18,-1
5070 DATA Courrier,27,120,1,-1,27,120,0,-1
5075 DATA Courrier prop,27,120,1,27,112,1,-1,27,
112,0,27,120,0,-1
5100 DATA Normal,-1,-1
5110 DATA Gras,27,69,-1,27,70,-1
5120 DATA Italique,27,52,-1,27,53,-1
```

```
5130 DATA Italique gras,27,52,27,69,-1,27,70,27,
53,-1
5140 DATA " ",-1,-1
5150 DATA Double frappe,27,71,-1,27,72,-1
5160 DATA Indice,27,83,1,-1,27,84,-1
5170 DATA Exposant,27,83,0,-1,27,84,-1
```

Voici la sortie qu'il produit en simple largeur. On constate qu'une combinaison "interdite" peut tout de même être utilisée, mais elle produit une impression déjà obtenue.

```
Simple
StandardNormal Double frappeIndiceExposant
StandardGras Double frappeIndiceExposant
StandardItalique Double frappeIndiceExposant
StandardItalique gras Double frappeIndiceExposant
MiniNormal Double frappeIndiceExposant
MiniGras Double frappeIndiceExposant
MiniItalique Double frappeIndiceExposant
MiniItalique gras Double frappeIndiceExposant
ProportionnelNormal Double frappeIndiceExposant
ProportionnelGras Double frappeIndiceExposant
ProportionnelItalique Double frappeIndiceExposant
ProportionnelItalique gras Double frappeIndiceExposant
CondenseNormal Double frappeIndiceExposant
CondenseGras Double frappeIndiceExposant
CondenseItalique Double frappeIndiceExposant
CondenseItalique gras Double frappeIndiceExposant
CourrierNormal Double frappeIndiceExposant
CourrierGras Double frappeIndiceExposant
CourrierItalique Double frappeIndiceExposant
CourrierItalique gras Double frappeIndiceExposant
Courrier propNormal Double frappeIndiceExposant
Courrier propGras Double frappeIndiceExposant
Courrier propItalique Double frappeIndiceExposant
Courrier propItalique gras Double frappeIndiceExposant
```

Nous montrons aussi ce que produisent les modes condensé et en double largeur :

```
Double largeur
CondenseNormal Double frappeIndiceExposant
CondenseGras Double frappeIndiceExposant
CondenseItalique Double frappeIndiceExposant
CondenseItalique gras Double frappeIndic
eExposant
CourrierNormal Double frappeIndiceExposa
nt

Souligne
StandardNormal Double frappeIndiceExposant
```

COMMANDES DE PARAMÉTRAGE

Dans cette catégorie, figurent des commandes d'installation de tabulations, de fixation des marges droite et gauche, de fixation de la longueur de page et, surtout, de fixation de l'interligne.

Nous n'étudions ici que les séquences de fixation de l'interligne :

- **27 48** fixe l'interligne à 1/8e de pouce.
- **27 49** fixe l'interligne à 7/72e de pouce (valeur qui convient en graphiques 7 aiguilles).
- **27 50** fixe l'interligne à 1/6e de pouce (valeur par défaut).
- **27 51 n** fixe l'interligne à n/216e de pouce.
- **27 65 n** fixe l'interligne à n/72e de pouce.

COMMANDES GRAPHIQUES

La DMP 2000 a plusieurs modes graphiques, de densités de points différentes. Nous n'en étudierons que deux correspondant aux séquences de contrôle :

27 75 n1 n2 d ... d et **27 76 n1 n2 d ... d**

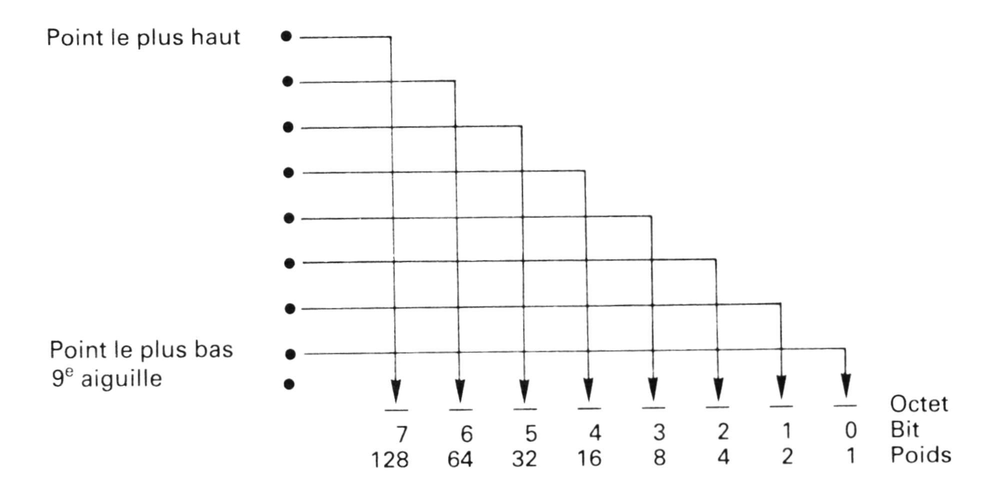
La seconde produit une densité horizontale de points double de la première (960 points par ligne contre 480).

Sinon, ces deux modes ressemblent beaucoup au mode graphique de la DMP 1 au moins en principe : n1 et n2 forment le nombre d'octets de données d qui spécifient chacune une colonne élémentaire à imprimer. Il y a quelques différences par rapport à la DMP 1 :

– n1 et n2 sont inversés : la partie basse du nombre n vient d'abord ;

– comme l'imprimante travaille en 8 bits, n=n1+256*n2. Il se pose alors une difficulté avec l'Amstrad qui a une interface 7 bits. Le plus simple est de ne fournir que des n inférieurs à 127 : vous pouvez toujours répéter la commande ;

– les données sont censées représenter 8 aiguilles disposées selon le schéma de la page ci-contre (là aussi, il y a inversion par rapport à la DMP1).



La difficulté de l'interface se repose à nous : le bit 7 sera toujours à 0 avec un Amstrad, donc l'aiguille du haut ne marquera jamais. De même la 9e aiguille n'est pas utilisable (il y a un mode qui l'utilise, mais il n'est pas accessible en 7 bits).

Nous devons donc travailler en 7 aiguilles. Ce n'est pas du tout gênant, car, si l'on prend la précaution de *régler l'interligne à 7/72e de pouce*, on obtient des lignes jointives.

Le WIDTH 255 est, lui, toujours nécessaire. On obtient alors le programme suivant qui est la version DMP 2000 de l'exercice 2.10. de copie d'écran graphique.

```
1 REM
3 REM Copie Ecran H-R DMP 2000
4 REM
10 WIDTH 255 :PRINT#8,CHR$(27);CHR$(49) : G$=CHR
$(27)+CHR$(75)+CHR$(100)+CHR$(0)
20 FOR X=0 TO 639 STEP 7
30 A$="" :B$="" :C$="" :D$="" : FOR Y=0 TO 99
40 N=0 : P=0 : M=0 : Q=0 : T=Y+100 : Z=Y+200 : U
=Y+300 : FOR I=0 TO 6
50 N=N-(2^I)*(TEST(X+6-I,Y)<>0)
52 P=P-(2^I)*(TEST(X+6-I,T)<>0)
55 M=M-(2^I)*(TEST(X+6-I,Z)<>0)
57 Q=Q-(2^I)*(TEST(X+6-I,U)<>0)
60 NEXT : A$=A$+CHR$(N) : C$=C$+CHR$(P) : B$=B$+
CHR$(M) : D$=D$+CHR$(Q)
70 NEXT
80 PRINT#8,G$;A$;G$;C$;G$;B$;G$;D$
90 NEXT
100 PRINT#8,CHR$(15)
```

Exercice 2.11. – Faire sur DMP 2000 le tracé de sinusoïde donné ci-dessus pour DMP 1. Pour changer, faites-le en double densité.

COMMANDES DIVERSES

On trouve d'abord dans cette catégorie les commandes d'avance de papier comme **10** (alimentation ligne), **12** (avance d'une page), **8** (recul d'un caractère) ou **13** (retour-chariot). Selon un interrupteur, le retour-chariot fait un simple retour à gauche permettant la superposition ou il fait en plus une avance de ligne : l'imprimante est livrée dans la première position.

On trouve aussi la sonnerie (**7**) et d'autres commandes spécialisées. Les commandes intéressantes de cette catégorie sont celles qui choisissent un jeu de caractères nationaux et celles liées aux caractères définis par l'utilisateur.

27 82 n

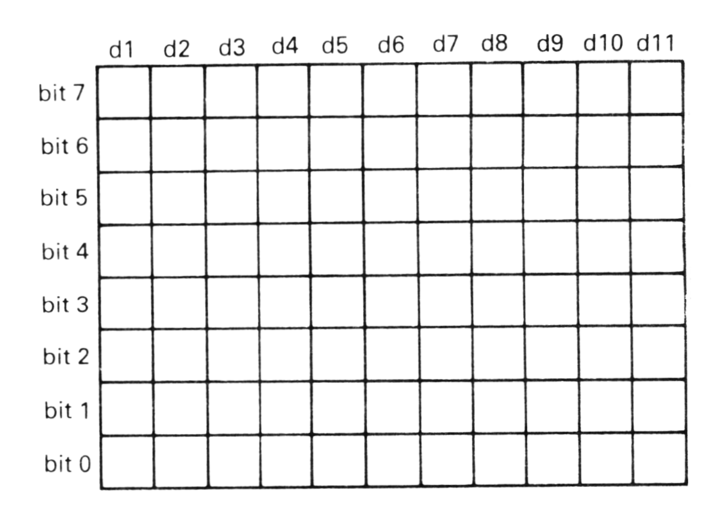
Cette séquence active le jeu de caractères nationaux n, quelle que soit la position des interrupteurs 1–1 à 1–3.

Les n à spécifier sont :

0 USA	1 France	2 Allemagne	3 GB	4 Danemark
5 Suède	6 Italie	7 Espagne	8 Japon	

CARACTÈRES DÉFINIS PAR L'UTILISATEUR

L'utilisateur peut définir le dessin des caractères de code 0 à 31 dans une grille 8×11 :



(Avec notre problème d'interface 7 bits, les bits 7 seront toujours 0.)

Pour définir une série de tels caractères, on doit d'abord envoyer une séquence :

27 38 0 premier dernier attr d1...d11 attr d1...d11 etc.

où premier et dernier définissent l'intervalle des codes à définir : en pratique, on ne définira qu'un caractère à la fois et donc, on prendra premier=dernier=code du caractère ; pour chaque caractère à définir, on a une série attr d1...d11, donc en pratique, on aura une seule série. Attr est un paramètre que nous laissons de côté, et, à ce moment, il faut lui donner la valeur 11 ; les d sont les données de la grille – attention, vous devez les agencer pour qu'il n'y ait pas de points à noircir consécutifs sur une même horizontale.

On doit ensuite dire qu'on veut utiliser le(s) caractère(s) défini(s) : c'est fait par la séquence 27 37 1 0 et, enfin, dire qu'on veut que les caractères de contrôle (en effet, les codes <30 sont des codes de contrôle) deviennent imprimables, ce que fait la séquence 27 73 1.

Il n'y a plus qu'à imprimer votre caractère par PRINT#8,CHR$(code).

Exemple

Le programme ci-dessous crée un alpha sous le code 5.

```
1 REM
2 REM ALPHA
3 REM
10 AL$=CHR$(5)
20 DE$="" :READ A : WHILE A<>-1
30 DE$=DE$+CHR$(A) :READ A : WEND
40 PRINT#8,DE$
50 PRINT#8,AL$;AL$
60 PRINT#8,AL$;AL$
100 DATA 27,38,0,5,5,11,28,0,34,0,34,0,28,0,34,0
,0,27,37,1,0,27,73,1,-1
```

Exercice 2.12. – On peut faire un alpha avec les commandes graphiques. Quel est l'avantage de la méthode ci-dessus ?

APPLICATION DE GESTION : LE PUBLIPOSTAGE (MAILING-LIST)

L'application "publipostage" (en anglais "mailing-list") consiste à imprimer une série d'adresses sur des étiquettes autocollantes présentées sur des liasses de papier au même format que le papier d'imprimante. Cela permet à

une entreprise, par exemple, d'envoyer une publicité à tous les clients de son fichier ou de faire des envois sélectifs.

Comme nous avons, sur cassette ou disquette, le fichier de nos amis à inviter, nous allons pouvoir leur envoyer nos cartes d'invitation.

En fait, le problème n'est pas très différent de ce que nous avons fait au chapitre précédent, lorsque nous avons listé nos amis sur l'écran.

Ce qui est différent, c'est que sur l'imprimante, il ne faut imprimer que le nom et l'adresse, et surtout, bien cadrer les libellés dans chaque étiquette.

Il faut imprimer les rubriques 1, 2, 3 et 4. La rubrique 3 doit éventuellement être imprimée en 2 lignes.

Pour faciliter le cadrage, il faut paramétrer au maximum les numéros de colonne de départ des différentes zones ainsi que les nombres de lignes blanches à laisser entre chaque impression. Il y a en effet une très grande diversité de formats d'étiquettes (largeur et hauteur) et de présentation : il existe des formulaires ayant de 1 à 4 étiquettes de front sur une rangée. Nous prendrons ici le cas de 1.

Ces paramètres sont donnés au clavier en réponse à des INPUT. Il faut quelquefois tâtonner un peu pour déterminer les meilleures valeurs à donner aux paramètres et faire attention au positionnement initial de l'imprimante.

Le résultat est le programme assez simplifié qui suit, pour lequel, outre le nom du fichier à utiliser, il faut fournir la largeur d'une étiquette, et les nombres de lignes à passer après le prénom, après l'adresse et à la fin de l'étiquette.

```
1 REM MAILING CASSETTE
10 DIM A$(10),B$(5)
20 INPUT "NOM DU FICHIER, Cass OU Disq";F$,P$
25 IF P$="C" THEN |TAPE ELSE |DISC
30 OPENIN F$
40 INPUT "LARGEUR, NB LIGNES A PASSER";LA,L1,L2,L3
45 S(1)=0:S(2)=L1:S(3)=0:S(4)=L2:S(5)=L3
50 INPUT#9,A$(1)
55 IF A$(1)="$$$" GOTO 140
60 FOR I=2 TO 10
65 INPUT#9,A$(I) : NEXT
70 B$(1)=LEFT$(A$(1),LA)
75 B$(2)=LEFT$(A$(2),LA)
80 B$(3)=LEFT$(A$(3),LA)
85 B$(4)=MID$(A$(3),LA+1,LA)
90 B$(5)=LEFT$(A$(4),LA)
```

```
100 FOR J=1 TO 5
110 PRINT#8,B$(J)
115 IF S(J)=0 GOTO 125
120 FOR M=1 TO S(J) : PRINT#8 : NEXT
125 NEXT
130 GOTO 50
140 PRINT "TERMINE" : CLOSEIN
```

Bien entendu, on pourrait varier la présentation : ne pas imprimer le prénom, mais mettre MONSIEUR, MADAME...

L'adresse est imprimée en deux lignes pour le cas où il y aurait un lieu-dit.

Exercice 2.13. – Faire un mailing sélectif, c'est-à-dire n'imprimer que les personnes qui satisfont à une condition.

CONCLUSION

Avec l'imprimante, nous nous sommes dotés d'un périphérique qui nous ouvre beaucoup d'applications. Notre système de fichiers s'est étoffé avec l'application mailing (et bien d'autres sont possibles). Il nous reste à nous affranchir de la lenteur des cassettes et de l'accès séquentiel (encore que notre système peut fonctionner tel quel sur disquettes). Le chapitre suivant est pleinement consacré aux disques.

CHAPITRE 3
LES DISQUETTES

INTRODUCTION – DESCRIPTION

C'est au début des années 60 que les disques durs sont apparus sur les gros ordinateurs. Ils apportaient l'avantage essentiel par rapport aux bandes magnétiques de permettre l'accès direct à l'information.

C'est exactement la même différence qu'il y a entre la bande de magnétophone et le disque : pour écouter le morceau de la fin sur une bande, vous devez dévider toute la bande alors que sur un disque, il vous suffit de déplacer le bras.

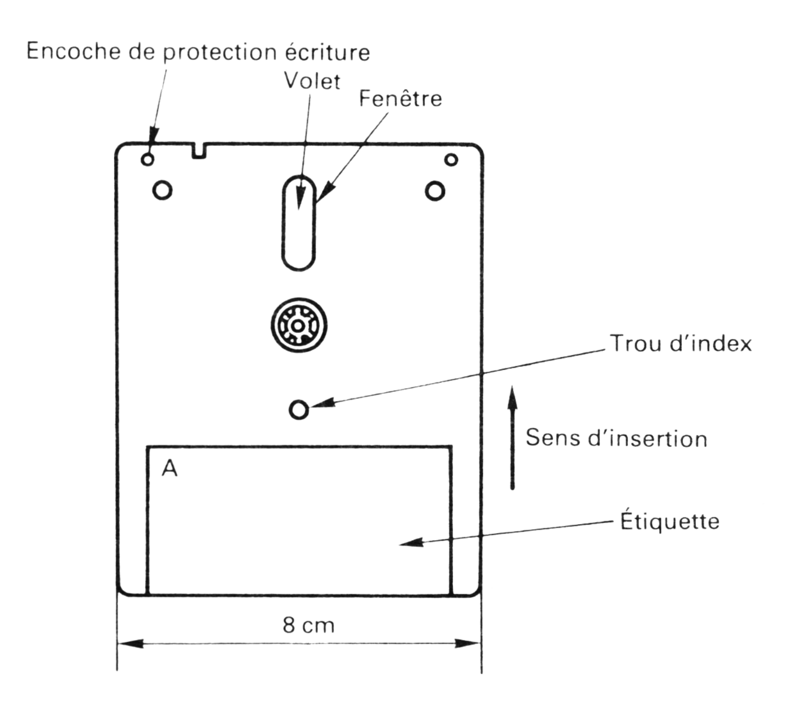
Mais l'analogie s'arrête là : sur les disques d'ordinateur, l'information est écrite magnétiquement sur des pistes concentriques et non gravée sur un sillon en spirale.

Les progrès de la technologie ont conduit à des capacités très élevées (on utilise des piles de plusieurs disques pour l'accroître), mais ces unités de disques constituaient tout de même des ensembles électromécaniques complexes de coût prohibitif pour les ordinateurs individuels. Notons toutefois que les micro-ordinateurs professionnels utilisent souvent des disques durs de technologie récente, de capacité modérée et dont la grande diffusion rend les prix abordables pour le marché professionnel.

Vers 1970, IBM introduisit un nouveau support à l'usage de ses terminaux de saisie de données ; le *disque souple*. De capacité bien moindre que les disques durs, le disque souple permet des performances plus faibles, mais il s'accommode d'unités plus simples et donc de coût compatible avec les micro-ordinateurs. Puis, à côté du disque souple (disquette) normal de diamètre 8 pouces (20 cm), est apparu le minidisque souple ou minidisquette de dia-

mètre 5,25 pouces (13 cm). Et maintenant, ce sont des disquettes de 3 pouces et 3,5 pouces que l'on trouve. Le format 3,5 pouces est très utilisé dans les ordinateurs portables professionnels tandis qu'Amstrad a choisi le standard 3 pouces (8 cm) ; il est regrettable qu'Amstrad soit un peu seul à avoir adopté cette norme.

La disquette est un disque de mylar (plastique résistant) recouvert d'oxyde magnétique et entouré d'une enveloppe carrée dans laquelle il tourne.



Mini-disque souple

L'enveloppe présente une fenêtre par laquelle la tête de lecture-écriture vient au contact de la surface magnétique pour lire ou écrire. Sur les 3 pouces, cette fenêtre est fermée par un volet lorsque la disquette est sortie de la platine. Aussi, l'enveloppe des trois pouces est rigide, ce qui est un avantage par rapport aux 5 et 8 pouces.

L'enveloppe comporte deux (une pour chaque face) encoches de protection écriture : si l'encoche est vide, l'écriture est impossible, ce qui protège la disquette contre toute altération intempestive des informations ; pour pouvoir écrire, il faut que l'encoche soit bouchée.

Enfin, un trou, dit *trou d'index*, permet à un couple émetteur-détecteur de lumière de déceler des trous percés à intervalles réguliers dans la disquette. Ces trous divisent chaque piste en *secteurs*, ce qui facilite la synchronisation permettant de relire les informations. Dans ce cas, la sectorisation est dite

matérielle (par hardware). La sectorisation peut également être *logicielle* (par software) : dans ce cas, il y a un seul trou qui marque le début de piste ; les secteurs ou blocs d'information sont délimités par des intervalles détectés par programme (ceci s'appelle le *formatage*).

Le formatage fait perdre un peu par rapport à la capacité brute de la disquette.

La capacité des disques peut être accrue de trois manières :

– doubler la densité d'information le long de chaque piste ;

– doubler le nombre de pistes concentriques rencontrées le long d'un rayon ;

– munir la platine tourne-disque de deux têtes ; le disque est alors utilisé simultanément sur ces deux faces.

L'Amstrad utilise un ersatz de cette dernière possibilité : les disquettes sont double-face, mais seule une face peut être utilisée à un instant donné ; il faut retourner la disquette pour utiliser l'autre face ; faites toujours bien attention à la face que vous utilisez.

Moyennant cela, les disquettes de l'Amstrad renferment environ 180 K octets par face. Avec d'autres normes, mais sur ordinateurs beaucoup plus chers, on fait beaucoup plus : jusqu'à 1 M octets par disquette.

Un autre paramètre est le temps d'accès. Retenons que, grossièrement, il faut compter 10 ms pour passer d'une piste à la suivante, et, qu'en moyenne, un accès comprenant le saut d'un certain nombre de pistes prend 100 ms.

Précautions d'emploi des disquettes

Un certain nombre de précautions vous permettront, si vous les suivez scrupuleusement, de prolonger la vie de vos disquettes (coût 30 à 60 F) et surtout d'éviter de perdre des informations précieuses, dont la reconstitution peut coûter des heures, voire des semaines de travail.

1. – Remettez toujours votre disquette dans sa boîte après usage.

2. – N'actionnez pas le mécanisme d'ouverture du volet : il ne sert à rien de regarder la surface magnétique de la disquette.

3. – Si vous le pouvez, évitez de fumer devant les disquettes : elles sont presqu'aussi fragiles que vos poumons. Evitez en tout cas absolument que des cendres ne viennent au contact de la surface d'enregistrement.

4. – Introduisez la disquette doucement dans la platine.

5. – Amstrad recommande de toujours allumer ou éteindre l'unité centrale et les unités de disques, les disquettes étant sorties. Cela évite tout risque d'altération. Toutefois, si vous avez allumé ou éteint en présence d'une disquette, ne soyez pas désespéré : la disquette sera probablement restée intacte.

Les unités de disque Amstrad

A part les unités incorporées au 664 et au 6128, il y a deux modèles d'unités de disques extérieures. Ce point mérite d'être clarifié car il y a près de 1 000 F de différence de prix.

La plus chère est à employer comme *première* unité sur un 464 : elle comprend la ROM qui fournit les commandes supplémentaires AMSDOS.

La moins chère est à employer comme *seconde* unité sur n'importe quel modèle.

Dans la suite, nous supposerons que vous n'avez qu'une unité. De toutes façons, la seconde s'utilise exactement comme la première, en précisant partout où il faut l'unité B au lieu de A.

PRINCIPES D'UTILISATION

La disquette est mise en œuvre à l'aide du système d'exploitation disque (SED). Avec l'Amstrad, c'est même deux SED que vous avez à votre disposition : AMSDOS et CP/M.

AMSDOS peut être considéré comme une extension à BASIC : d'ailleurs ses commandes peuvent être fournies soit en mode direct soit à l'intérieur d'un programme BASIC. Elles commencent toutes par le caractère ¦ (SHIFT @).

CP/M est un SED beaucoup plus élaboré que AMSDOS. C'est d'ailleurs un standard du marché et de nombreux logiciels tout faits existent sous CP/M. Il offre de nombreuses commandes et utilitaires. Nous n'étudierons que les plus fondamentales. Certaines sont indispensables car elles n'ont pas leur équivalent sous AMSDOS, notamment le formatage et la copie d'une disquette.

Le monde des Amstrad connaît deux versions de CP/M : CP/M 2.2 livré avec le 664 ou avec l'unité connectable au 464 et CP/M Plus (autre nom de la version 3) livré avec le 6128. CP/M Plus est plus élaboré que CP/M 2.2, mais

nous exploiterons peu les différences dans ce livre. CP/M Plus permet notamment de redéfinir facilement le clavier et de configurer certains périphériques.

Avec le 664 (ou l'unité connectable au 464), vous recevez une disquette qui contient sur la face 1 CP/M 2.2 et ses utilitaires et sur la face 2 le langage Logo.

Avec le 6128, vous recevez 2 disquettes, soit 4 faces qui contiennent :

- face 1 : CP/M Plus avec des utilitaires notamment BANKMAN ;
- face 2 : utilitaires comme ASM, LINK et programmes graphiques ;
- face 3 : utilitaires graphiques, HELP et Logo version 3 ;
- face 4 : CP/M 2.2 et Logo version 2.2.

La présence de CP/M version 2.2 vous permet de récupérer des programmes préparés sous CP/M 2.2.

COMMANDES CP/M FONDAMENTALES

Dans ce chapitre, nous ne verrons que les deux commandes CP/M qui sont indispensables dès que l'on utilise les disques. Nous verrons les autres au chapitre 4. Nous verrons aussi quelques commandes AMSDOS dans ce chapitre.

Duplication d'une disquette

Il est *impératif, avant tout travail sur disquette,* de dupliquer la (les) disquette(s) système et vous devez travailler avec les copies, les originaux étant mis de côté. Libre à vous de ne pas obéir à cette précaution, mais sachez que les disquettes sont fragiles et qu'il y a des risques de fausses manœuvres, surtout pour les débutants et même alors que les disquettes système sont livrées protégées en écriture. Même si, en cas d'accident, votre revendeur consent à vous remplacer pour un coût minime les disquettes système, vous aurez à attendre un délai bien plus long que le temps de la duplication effectuée préventivement. De plus, effectuer ces duplications vous habitue à l'utilisation des disquettes.

Il n'y a pas de commande de duplication sous AMSDOS. Il faut donc passer sous CP/M. A la mise sous tension, vous êtes sous AMSDOS. Vous passez sous CP/M par la commande AMSDOS :

```
¦CPM
```

Sur 464/664, le fond devient bleu clair et l'on passe en 80 colonnes sur l'écran. Il n'y a pas de changement de couleur sur 6128 avec CP/M Plus.

Notons dès maintenant que l'on repasse sous AMSDOS grâce à la commande (CP/M cette fois) :

AMSDOS

Vous aurez à le faire lorsque, ayant terminé les duplications, vous voudrez repasser à BASIC.

Sous CP/M 2.2 (donc sur 464 et 664), vous invoquez la commande DISC-COPY en tapant simplement ce mot au clavier. Cette commande est adaptée au cas où vous n'avez qu'une unité de disquettes : elle vous demandera à plusieurs reprises d'échanger dans la platine la disquette source (celle que vous copiez) et la disquette destination (la copie en train de se faire). Si vous avez deux unités, utilisez plutôt COPYDISC qui évite les montages et démontages de disquettes.

Sous CP/M Plus (donc sur 6128), vous invoquez un programme utilitaire en tapant son nom : DISCKIT3. Une menu apparaît alors. Vous choisissez la fonction COPY en appuyant sur la touche **f7** du clavier numérique.

Le système "sait" si vous avez une ou deux unités. Selon le cas, les messages sont différents, mais ils sont très clairs et expliqués dans le manuel constructeur. Ceci vaut d'ailleurs pour les deux versions.

Nous supposerons que votre copie s'est bien passée. Vous devez la faire au moins une fois pour chaque face, donc il y a deux faces à copier sur 464/664 et quatre sur 6128. Vous pouvez copier sur la même disquette les deux faces d'une même disquette originale ; attention à ne pas vous tromper de face !

Pendant que nous y sommes et puisque la duplication demande de nombreux montages et démontages de disquettes, rappelons comment s'effectuent ces manœuvres. Pour monter une disquette, il suffit de l'enfoncer doucement mais à fond dans la fente de la platine ; la lettre de la face utilisée doit être vers le haut. Pour la démonter, il suffit d'appuyer sur le bouton situé sur la façade de la platine : elle sort automatiquement.

Formatage

Une autre commande fondamentale est celle de formatage. Toute disquette neuve doit subir cette opération avant de pouvoir être utilisée. En effet, en sectorisation logicielle, il faut inscrire le long des pistes des marques qui

permettront ultérieurement de se repérer sur la disquette. Le formatage est effectué automatiquement lors d'une commande de duplication.

Vous pouvez faire subir un formatage à une disquette déjà utilisée, mais, attention, toutes les anciennes informations se trouveront effacées.

Vous pouvez ne jamais utiliser le formatage et constituer toutes vos faces de disquettes par duplication de la première face système. Ainsi, vous aurez toujours le système et ses utilitaires, ce qui peut être intéressant si vous n'avez qu'une unité. Toutefois, cela revient à réduire la place disponible pour vos données sur chaque disquette.

Les différents formats possibles sont :

– (pour mémoire) système + utilitaires (obtenu par duplication) ;

– s (système) : laisse 169 K octets libres ;

– d (données) : laisse 178 K octets libres ;

– v (vendeur) : laisse 169 K libres ; le système est absent mais sa place est réservée. Ce format est utile pour un vendeur de logiciel qui n'a pas le droit de "vendre" CP/M : il livre son logiciel sur une disquette au format v et l'acheteur peut y copier sa version de CP/M.

– i (IBM) : laisse 154 K octets libres ; ce format est "compatible" IBM, mais CP/M Plus ne le permet plus. Il a donc peu d'utilité.

Sous CP/M 2.2, vous tapez la commande FORMAT lettre où 'lettre' est l'option de formatage choisie, par exemple FORMAT D. On vous demande ensuite de monter la disquette à formater.

Sous CP/M Plus, vous utilisez l'option format de DISCKIT3. Vous avez ensuite à choisir l'option de formatage. Rappelons que l'option IBM est absente. Les deux options vraiment fréquentes sont 's' ou 'd'.

Rappelons enfin que vous devez effectuer l'opération pour chaque face de la disquette.

Notons que ces deux commandes sont ce qu'on appelle des "commandes externes" de CP/M : pour qu'elles fonctionnent, il faut que soient présents sur la disquette, non seulement le système, mais aussi les fichiers DISCKIT3.COM ou FORMAT.COM et DISCCOPY.COM.

Visualisation du contenu d'une disquette

CP/M et AMSDOS ont chacun une commande pour cette fonction ; il est vrai qu'elle est essentielle.

Sous AMSDOS, on utilise la commande CAT (à condition que ce soit la disquette qui soit active : faites éventuellement un ¦DISC).

Sous CP/M, on utilise la commande DIR éventuellement suivie du nom de l'unité de disquette à mettre en jeu et de ':', par exemple, DIR B :

La liste ne se présente pas tout à fait de la même façon selon la commande utilisée. Avec CAT, on obtient aussi l'encombrement de chaque fichier et la place restante sur le disque.

Avec DIR, on a un autre avantage, dû à une particularité de CP/M. Sous CP/M, certaines touches du clavier reçoivent des fonctions particulières. Il y en a plus sous CP/M Plus que sous CP/M 2.2, mais dans tous les cas, la combinaison **CTRL P** dirige vers l'imprimante toutes les sorties qui iraient sinon vers l'écran. Par conséquent, pour avoir le contenu d'une disquette sur imprimante, il suffit de faire :

¦CPM (pour passer sous CP/M à partir d'AMSDOS)
CTRL P (pour activer l'imprimante)
DIR

Si vous voulez revenir au mode écran, faites à nouveau **CTRL P**.

On obtient un listing tel que celui qui apparaît ci-dessous :

```
dir
A: MOVCPM   COM : PIP      COM : SUBMIT   COM : XSUB     COM
A: ED       COM : ASM      COM : DDT      COM : LOAD     COM
A: STAT     COM : DUMP     COM : DUMP     ASM : AMSDOS   COM
A: FILECOPY COM : SYSGEN   COM : BOOTGEN  COM : COPYDISC COM
A: CHKDISC  COM : DISCCOPY COM : DISCCHK  COM : SETUP    COM
A: FORMAT   COM : CSAVE    COM : CLOAD    COM : EX1      BAS
A: EX2      BAS : ROINTIME DEM : RITDEMO  BIN : DISC     BAS
A: RS232B   BAS : RS232MC  BIN
```

Nous n'avons pas plus à dire sur les noms de fichiers que ce qui a été dit au chapitre 1. La liste ci-dessus confirme la forme nom.extension (le point ne figure pas sur la liste). On voit beaucoup d'extensions .COM : rappelons que cela désigne des fichiers programmes exécutables par simple appel de leur nom sous CP/M. *Exemple :* FORMAT. .BAS désigne des fichiers BASIC : pour les exécuter, il faut faire RUN "nom", *exemple :* RUN "DISC". .BIN désigne un fichier binaire, .ASM un fichier assembleur.

Voyons maintenant les commandes AMSDOS.

COMMANDES AMSDOS

Parmi les commandes AMSDOS, figurent en fait toutes les instructions BASIC relatives aux fichiers que nous avons vues au chapitre 1 à propos des cassettes. Ce que nous voyons ici, ce sont les commandes proprement AMSDOS. Elles commencent par le caractère ¦. Comme les autres (OPENIN,

PRINT#, etc.), elles peuvent s'employer en mode direct ou à l'intérieur d'un programme BASIC.

Nous en avons déjà vu quelques-unes, comme ¦CPM ou ¦TAPE.

¦A

Dirige les entrées-sorties sur la disquette A. N'a d'intérêt que si vous avez deux disquettes.

¦B

Dirige les entrées-sorties sur la disquette B. N'a d'intérêt que si vous avez deux disquettes.

¦CPM

Passe sous CP/M.

¦DIR

Fournit la table des matières de la disquette de façon intermédiaire entre CAT et DIR de CP/M : les fichiers sont listés comme par DIR mais on a en plus la place restant libre sur la disquette.

On a aussi la forme DIR,"nom" où nom est une spécification de fichier. On obtient alors les renseignements sur les fichiers compatibles avec la spécification (voir plus loin "Spécification ambiguë des fichiers").

¦DISC

Dirige les entrées-sorties sur disque. C'est la commande contraire de ¦TAPE.

¦DISC.IN

Dirige les entrées sur disque, les sorties restant sur cassette. C'est le contraire de ¦TAPE.IN.

¦DISC.OUT

Dirige les sorties vers le disque, les entrées restant sur cassette. C'est le contraire de ¦TAPE.OUT.

¦DRIVE,"d"

Où d est A ou B : dirige les entrées-sorties vers l'unité de disques A ou B. N'a d'intérêt que si vous avez deux platines.

¦ERA,"nom"

Efface le fichier de nom indiqué, ou tous les fichiers compatibles si la spécification est ambiguë.

¦REN,"nom1","nom2"

Change le nom du fichier "nom2" : ce sera désormais "nom1".

¦TAPE

Dirige les entrées-sorties sur cassette.

¦TAPE.IN

Dirige les entrées sur cassette, les sorties restant sur disquette.

¦TAPE.OUT

Dirige les sorties sur cassette, les entrées restant sur disquette.

¦USER,numéro

Définit un numéro d'utilisateur. Les fichiers sont regroupés par utilisateurs, les accès aux fichiers d'un autre utilisateur étant restreints. Ceci n'a aucun intérêt sur un ordinateur individuel. Gardez donc le numéro 0 qui est considéré par défaut.

SPÉCIFICATION AMBIGUË DES FICHIERS

On peut désigner plusieurs fichiers d'un seul coup à l'aide des caractères spéciaux ? et *.

? signifie "n'importe quel caractère à ma position".

Exemple

PRO? est compatible avec PROG, PROF, PROU, etc., mais pas avec PROFIT.

* signifie "n'importe quelle série de caractères, même vide, à partir de ma position".

Exemple

PRO* est compatible avec PRO tout court, PROF, PROG, PROFIT, PROGRAMM, etc., mais pas avec PRUNE ni PROG.BAS. Il faut en effet fournir les ? et * pour l'extension s'il y a lieu.

Exemple

* signifie "n'importe quel nom sans extension"
** signifie "n'importe quel fichier avec ou sans extension".

Exercice 3.1. – Que signifient :

B:PROG.*
*.BAS
P*.B*

Exercice 3.2. – Désigner tous les fichiers ayant ZOZO comme caractères n° 4 à 7.

La désignation ambiguë existe sous AMSDOS et sous CP/M. Son effet diffère avec les commandes où elle intervient.

En principe, elle ne peut s'employer pour des opérations d'écriture (*ex.* OPENOUT, ¦REN...).

Pour une opération de lecture comme OPENIN, elle agira sur le *premier* fichier compatible trouvé sur la disquette.

Pour une opération de table des matières (DIR ou ¦DIR), elle fournira *tous* les fichiers compatibles avec la désignation. Par exemple, si nous voulons la liste de tous les fichiers dont le nom commence par X, il suffit de faire :

¦DIR,"X*.*"

Il en est de même – *et c'est très dangereux* – pour une opération d'effacement. ¦ERA,"Z*" ou, sous CP/M, ERA Z* efface tous les fichiers dont le nom commence par Z. Tant pis pour vous si vous vouliez garder ZOZO ! Sous CP/M Plus, vous avez une légère rémission, avec l'option CONFIRM qui fait demander confirmation avant chaque effacement. Toutefois, c'est à éviter.

L'utilitaire CP/M 2.2 de copie de fichiers FILECOPY a lui aussi l'interprétation "tous", mais il demande confirmation à chaque copie.

OPÉRATIONS DE LECTURE ET ÉCRITURE

Les opérations de sauvegarde et chargement de programmes sur disquette sont exactement les mêmes que sur cassette et elles ont donc été traitées au chapitre 1.

Il en est de même des opérations de lecture et écriture sur fichier séquentiel. Nous n'avons rien à ajouter par rapport au chapitre 1.

Ce que les disques ajoutent, en fait, c'est la possibilité d'accès direct, c'est-à-dire la possibilité de dire à *quel enregistrement* du fichier l'on veut accéder.

Malheureusement, le BASIC Amstrad ne possède pas d'instruction permettant l'accès direct. Nous allons donc, dans les paragraphes qui suivent, *simuler* cet accès direct.

SIMULATION D'UN SYSTÈME DE FICHIER EN ACCÈS DIRECT

L'idée de base de notre simulation est la suivante : chaque enregistrement de notre fichier constituera lui-même un fichier. On parlera donc de fichier global décomposé en fichiers partiels. Le nom d'un fichier partiel sera formé par concaténation du nom de fichier choisi et du numéro de fichier partiel. Si par exemple l'on veut traiter un fichier d'adresses, les noms des fichiers partiels seront ADRES1, ADRES2, etc.

Ceci résout totalement le problème de l'accès direct. Si l'on désire lire l'enregistrement n° N, il suffit de faire :

N$=MID$(STR$(N),2) (pour éliminer l'espace en tête) et
OPENIN "ADRES"+N$

Le problème de l'impossibilité d'avoir plusieurs fichiers simultanément est résolu aussi : pour chaque fichier, on lit l'enregistrement dont on a besoin et l'on referme aussitôt.

L'ennui est que cela ne peut pas marcher à cause d'une autre limitation du système-disque de l'Amstrad. En effet, il ne peut y avoir plus de 64 fichiers sur une disquette (ou plutôt une face) ; c'est dû tout simplement à la limite de taille du catalogue.

Il en résulte donc que notre fichier ainsi décomposé en enregistrements-fichiers partiels ne pourrait renfermer que 64 enregistrements, donc ne porter

que sur au plus 64 individus. C'est une limitation inacceptable. Comment la dépasser ?

La solution est que les fichiers partiels contiennent plusieurs enregistrements, soit NG ce nombre. On aura donc en mémoire à un instant donné une "tranche" de NG enregistrements du fichier global. Ces enregistrements forment, sur disque, un fichier partiel et résident en mémoire dans un tableau de chaînes de caractères ENR$ (il est préférable de traiter toutes les informations comme des chaînes de caractères). Un enregistrement a deux numéros, E, son numéro dans le fichier global et EE, son numéro dans son fichier partiel ; lorsque son fichier partiel réside en mémoire, cet enregistrement est ENR$(EE). Soit F le numéro de fichier partiel d'un enregistrement. Pour trouver l'enregistrement n° E, il faut trouver les valeurs de EE et F correspondant à E. C'est facile, connaissant NG : F est grosso-modo le quotient de E par NG et EE est le reste de cette division.

Plus précisément, on a :

EE=1+(E−1) MOD NG

et F=1+INT((E−1)/NG)

Maintenant, l'enregistrement ENR$(EE) est formé de rubriques. Pour la rubrique n° R, on connaît son libellé (ou sa désignation) LIB$(R), le n° de caractère où elle commence dans ENR$(EE) D(R) et sa taille (nombre de caractères) TA(R).

La rubrique n° R de l'enregistrement E est alors la chaîne de caractères R$=MID$(ENR$(EE),D(R),TA(R)).

Ces variables sont l'essentiel ; avec quelques autres, on arrive à la table des variables suivante pour les fonctions de création du fichier et d'ajout d'enregistrements :

NOM$	Nom du fichier (ramené à 6 caractères).
NR	Nombre de rubriques.
NEP	Nombre d'enregistrements prévus.
LIB$(R)	Libellé de la rubrique R.
TA(R)	Taille (nombre de caractères) de la rubrique R.
D(R)	Début (n° du premier caractère) de la rubrique R.
R$	Contenu de la rubrique R : R$=MID$(ENR$(EE),D(R),TA(R)).
TE	Taille d'enregistrement (somme des TA).
ME	Taille mémoire pour un fichier partiel. Valeur typique : 16000 octets.
NG	Nombre d'enregistrements présents en mémoire=nombre d'enregistrements d'un fichier partiel.
ENR$(EE)	Enregistrement n° EE présent en mémoire. Le tableau ENR$ est dimensionné à NG.
EE	N° d'enregistrement en mémoire ou dans son fichier partiel.

E	N° d'enregistrement dans le fichier global.
F	N° du fichier partiel où se trouve un enregistrement.
FP	Valeur précédente de F.
F$	Forme chaîne de caractères de F (dépourvue de son espace en tête).
NE	Nombre d'enregistrements effectivement écrits.

A côté des fichiers partiels, nous aurons à gérer un fichier fondamental, que nous appelons le "fichier résumé". En fait, il contiendra les paramètres fondamentaux du fichier : nombre de rubriques, libellés et tailles des rubriques, etc. Il contient aussi le nombre d'enregistrements écrits (0 à la création du fichier). Toute utilisation du fichier global doit consulter le fichier résumé. On lui donnera le nom choisi comme nom de la base de données (rappelons que les fichiers partiels ont pour nom ce nom suivi du numéro de fichier partiel).

La création se fait en deux étapes :

1. – Création du fichier résumé – étape appelée par RUN 1000 –. En réponse aux questions convenables, on fournit les paramètres fondamentaux du fichier.

2. – Ajout d'enregistrements – étape appelée par RUN 2000 et répétable autant de fois qu'on veut –. En réponse aux questions, on fournit cette fois des données effectives pour le fichier. Lorsqu'on veut arrêter, on fournit $$$ comme premier champ d'un enregistrement.

On trouvera ci-dessous le listing correspondant à ces deux fonctions. Les sous-programmes utilisés sont :

- 10000 : Ecriture du fichier résumé.
- 10500 : Remplissage du tableau ENR$ avec des caractères $.
- 11000 : Lecture du fichier résumé.
- 12000 : Lecture d'un bloc en mémoire. En fait ce sous-programme a pour rôle de retrouver l'enregistrement E. Il calcule le F et le EE correspondants. Si le fichier partiel qui contient l'enregistrement est déjà présent (F=FP), il n'y a rien d'autre à faire ; sinon, le fichier partiel correspondant est lu en mémoire.
- 13000 : Acquisition d'un enregistrement par réponse aux questions.
- 14000 : Ecriture sur disque du fichier partiel qui vient d'être rempli en mémoire.

```
1000 REM CREATION BASE DE DONNEES
1001 REM
1010 INPUT "Nom du fichier a creer";NOM$ : NOM$=
LEFT$(NOM$,6)
1020 INPUT "Nombre de rubriques, Nombre d'enregi
strements prevus";NR,NEP
1030 DIM LIB$(NR),TA(NR),D(NR)
```

```
1040 FOR R=1 TO NR :PRINT :PRINT "Rubrique No";R
 :PRINT "Libelle, Taille";
1050 INPUT LIB$(R),TA(R) :NEXT
1060 D(1)=1 : FOR R=2 TO NR
1070 D(R)=D(R-1)+TA(R-1) : NEXT
1080 TE=D(NR)+TA(NR)-1
1090 ME=80
1100 NG=INT(ME/TE)
1110 DIM ENR$(NG) : CR$=CHR$(13)
1120 IF NEP/NG>30 THEN PRINT "Nb enregistrements
 prevu trop grand; Ne depassez pas ";30*NG
1130 NE=0 : GOSUB 10000 ' Ecriture Fichier Resum
e
1140 END
1999 REM
2000 REM AJOUT
2001 REM
2010 INPUT "Nom du fichier";NOM$ : NOM$=LEFT$(NO
M$,6)
2020 GOSUB 11000 ' Lecture Fichier Resume
2030 GOSUB 10500 ' Remplissage avec $
2040 FP=0
2050 IF NE=0 THEN E=1:F=1:F$="1":EE=1:GOTO 2080
2060 E=NE+1
2070 GOSUB 12000 ' Lecture Bloc en memoire
2080 GOSUB 13000 ' Acquisition Enregistrement
2090 IF LEFT$(R$,3)="$$$" GOTO 2160
2100 E=E+1 : NE=NE+1
2110 EE=EE+1 : IF EE<=NG GOTO 2080
2120 GOSUB 14000 ' Ecriture Bloc
2130 GOSUB 10500 ' Remplissage avec $
2140 F=F+1 : EE=1 : F$=MID$(STR$(F),2)
2150 GOTO 2080
2160 GOSUB 14000 ' Ecriture Bloc
2170 GOSUB 10000 ' Ecriture Fichier Resume
2180 PRINT NE;" Enregistrements ecrits"
2190 END

9999 REM
10000 REM ECRITURE FICHIER RESUME
10001 REM
10010 OPENOUT NOM$
10020 PRINT#9,NR;CR$;NEP;CR$;
10030 FOR R=1 TO NR
10040 PRINT#9,LIB$(R);CR$;TA(R);CR$;D(R);CR$;
10050 NEXT
10060 PRINT#9,TE;CR$;ME;CR$;NG;CR$;NE;CR$;
10070 CLOSEOUT
10080 |ERA,"*.bak"
```

```
10090 RETURN
10499 REM
10500 REM REMPLISSAGE DE ENR$ AVEC DES $
10501 REM
10510 FOR G=1 TO NG
10520 ENR$(G)=STRING$(TE,"$") : NEXT
10530 RETURN
10999 REM
11000 REM LECTURE FICHIER RESUME
11001 REM
11010 OPENIN NOM$
11020 INPUT#9,NR
11030 INPUT#9,NEP
11040 DIM LIB$(NR),TA(NR),D(NR)
11050 FOR R=1 TO NR
11060 INPUT#9,LIB$(R)
11070 INPUT#9,TA(R)
11080 INPUT#9,D(R)
11090 NEXT
11100 INPUT#9,TE
11110 INPUT#9,ME
11120 INPUT#9,NG
11130 INPUT#9,NE
11140 DIM ENR$(NG) : CR$=CHR$(13)
11150 CLOSEIN
11160 RETURN
11999 REM
12000 REM LECTURE D'UN FICHIER PARTIEL
12001 REM
12010 EE=1+(E-1) MOD NG
12020 F=1+INT((E-1)/NG)
12030 F$=MID$(STR$(F),2)
12040 IF F=FP THEN RETURN
12050 FP=F
12060 OPENIN NOM$+F$
12070 FOR G=1 TO NG
12080 INPUT#9,ENR$(G) : NEXT
12090 CLOSEIN
12100 RETURN
12999 REM
13000 REM ACQUISITION D'UN ENREGISTREMENT
13001 REM
13010 PRINT "Enregistrement No ";E
13020 FOR R=1 TO NR
13030 PRINT LIB$(R); : INPUT R$
13040 IF LEFT$(R$,3)="$$$" THEN R=NR : GOTO 1307
0
13050 R$=LEFT$(R$+"                    ",TA(R))
13060 MID$(ENR$(EE),D(R),TA(R))=R$
```

```
13070 NEXT
13080 RETURN
13999 REM
14000 REM ECRITURE D'UN FICHIER PARTIEL
14001 REM
14010 OPENOUT NOM$+F$
14020 FOR G=1 TO NG
14030 PRINT#9,ENR$(G);CR$; : NEXT
14040 CLOSEOUT
14050 |ERA,"*.bak"
14060 RETURN
```

Les deux sous-programmes d'écriture se terminent par un effacement de tout fichier .BAK : en effet, si le fichier écrit existe déjà, il en est conservé une sauvegarde ; notre effacement évite l'encombrement de la disquette par ces sauvegardes. Si le fichier n'existe pas, il y a un message d'erreur dont il ne faut pas se soucier.

On remarquera en 1090 la valeur très faible que nous donnons à ME ; c'est pour la vérification de notre programme : nous créons un fichier très petit ; il faut que, malgré cela, il fasse intervenir plusieurs fichiers partiels, sinon la vérification du programme serait incomplète.

Quelle valeur adopter en utilisation réelle ? Nous pensons à une valeur voisine de 16000 (zone de 16 K octets). Cela permet de mettre en œuvre simultanément deux fichiers gérés de cette manière : il faut, bien sûr, que toutes les variables fondamentales soient doublées ; on aura notamment ENR1$ et ENR2$.

Supposons que chaque enregistrement occupe 200 octets. On aura alors NG=80 enregistrements par fichier partiel. Avec la limite sur le catalogue, et si l'on veut gérer deux fichiers globaux, on pourra avoir 30 fichiers partiels, soit un fichier de 30×80=2400 individus, ce qui est peu, mais dépasse déjà de beaucoup la capacité des disquettes de l'Amstrad (480 K contre 180 K). L'Amstrad n'est donc pas capable de traiter des applications fichiers professionnelles. Mais, avec le système que nous donnons, les applications individuelles deviennent faciles.

Recherche en clair

La première utilisation que nous ferons de notre base de données sera de rechercher et d'imprimer tous les enregistrements qui satisfont à une condition donnée. La condition sera donnée en clair, c'est-à-dire sous la forme d'une chaîne de caractères telle que NOM=DUPONT, si le fichier comporte une rubrique de libellé NOM.

N.B. – Il est très important que vous suiviez réellement le fonctionnement de ces programmes en constituant effectivement un petit fichier sur disquette. Prenez par exemple votre carnet d'adresses. Pour l'étape de création, vous décidez vous-même des rubriques à adopter et de leur taille. Vous pouvez prendre les mêmes paramètres qu'au chapitre 1 (sauf que toutes les zones sont ici traitées en alphanumérique.

Le listing figure ci-dessous. De 3040 à 3190, une routine très importante décompose la chaîne condition en ses éléments : le libellé de la rubrique concernée, l'opérateur de comparaison utilisé (on peut utiliser < > = <= >= et <>) et la valeur de comparaison. De 3200 à 3240, on cherche à quel numéro de rubrique correspond le libellé demandé. La fin du programme est la boucle de parcours de tous les enregistrements du fichier (on a ici une lecture séquentielle) et d'impression des enregistrements qui satisfont à la condition de sélection.

On a inclus aussi la possibilité de liste inconditionnelle du fichier : il suffit de spécifier $ comme condition.

Les sous-programmes nouveaux qui s'introduisent sont :

15000 : Test pour savoir si la condition est satisfaite (SAT=1) pour un enregistrement.

16000 : Impression d'un enregistrement sur écran ou sur imprimante selon la réponse à la question posée en 3020.

16500 : Si la sortie se fait sur écran, avant de passer à un nouvel enregistrement, on demande à l'utilisateur s'il autorise de passer à la suite. Si on ne le faisait pas, le déroulement continu de l'écran interdirait la lecture.

```
2999 REM
3000 REM LISTE SELECTIVE OU NON
3001 REM
3010 INPUT "Nom du Fichier";NOM$ : NOM$=LEFT$(NO
M$,6)
3020 INPUT "Sortie sur Imprimante - oui/non";X$
: IF X$="oui" OR X$="OUI" THEN IMP=8 ELSE IMP=0
3030 GOSUB 11000 ' Lecture Fichier Resume
3040 INPUT "Condition ($ pour pas de condition)"
;CND$
3050 IF CND$="$" GOTO 3250
3060 IC=0:IV=0:RU$="":CC$="":VV$=""
3070 FOR I=1 TO LEN(CND$)
3080 A$=MID$(CND$,I,1)
3090 IF A$=" " GOTO 3170
3100 IF A$="<" OR A$="=" OR A$=">" GOTO 3130
3110 IF IC>0 GOTO 3150
3120 RU$=RU$+A$ : GOTO 3170
```

```
3130 IC=IC+1 : IF IC>2 GOTO 3190
3140 CC$=CC$+A$ : GOTO 3170
3150 IF IC=0 GOTO 3190
3160 IV=IV+1 : VV$=VV$+A$
3170 NEXT
3180 IF IC*IV<>0 GOTO 3200
3190 PRINT "Erreur sur la Condition" : END
3200 FOR RR=1 TO NR
3210 IF LIB$(RR)=RU$ GOTO 3240
3220 NEXT
3230 PRINT "Rubrique ";RU$;" non trouvee" : END
3240 VV$=LEFT$(VV$+"                              ",TA(RR))
3250 FP=0 : FOR E=1 TO NE
3260 GOSUB 12000 ' Lecture d'un Bloc
3270 IF CND$="$" GOTO 3310
3280 R$=MID$(ENR$(EE),D(RR),TA(RR))
3290 GOSUB 15000 ' Verification Condition
3300 IF SAT=0 GOTO 3320
3310 GOSUB 16000 ' Sortie enregistrement
3320 NEXT
3330 END


14999 REM
15000 REM TESTE CONDITION
15001 REM
15010 SAT=0
15020 FOR C=1 TO IC
15030 IF (MID$(CC$,C,1)="<") AND (R$<VV$) THEN S
AT=1
15040 IF (MID$(CC$,C,1)="=") AND (R$=VV$) THEN S
AT=1
15050 IF (MID$(CC$,C,1)=">") AND (R$>VV$) THEN S
AT=1
15060 NEXT
15070 RETURN
15999 REM
16000 REM IMPRIME ENREGISTREMENT
16001 REM
16010 PRINT#IMP : PRINT#IMP
16020 PRINT#IMP,"Enregistrement No ";E;"  (";F;E
E;" )"
16030 PRINT#IMP
16040 FOR R=1 TO NR
16050 PRINT#IMP,LIB$(R),MID$(ENR$(EE),D(R),TA(R)
)
16060 NEXT
16070 IF IMP=0 THEN GOSUB 16500 ' Attente Touche
16080 RETURN
```

```
16499 REM
16500 REM ATTEND UN APPUI SUR UNE TOUCHE AVANT D
E DEROULER L'ECRAN
16501 REM
16510 PRINT : PRINT
16520 PRINT"Appuyez sur une touche quelconque po
ur  la suite"
16530 A$=INKEY$ : IF A$="" GOTO 16530
16540 RETURN
```

Les principales variables qui s'introduisent sont :

IMP	Périphérique d'impression (0=écran, 8=imprimante).
CND$	Chaîne condition ($ si liste inconditionnelle).
RU$	Libellé de la rubrique concernée dans la condition.
CC$	Opérateur de comparaison dans la condition.
IC	Longueur de CC$.
VV$	Valeur de comparaison dans la condition.
IV	Longueur de VV$.
RR	Numéro de la rubrique concernée.
SAT	=1 si la condition est satisfaite, 0 sinon.

Exercice 3.3. – Quelles modifications effectuer pour que le programme admette des conditions de la forme libellé#chaîne, satisfaites si la rubrique de libellé indiqué *contient* la chaîne donnée.

Exercice 3.4. – Il est fastidieux d'avoir à invoquer les différentes fonctions du programme par des RUN 1000, RUN 2000, etc. Modifiez le programme pour qu'il ait une en-tête de type menu.

Tri

Le second programme fait, lui, appel à l'accès direct. Il permet de lister le fichier dans l'ordre voulu en fonction de l'une des rubriques, par exemple ordre alphabétique des noms, notes décroissantes, montant des dettes croissant, etc. Le programme est en quatre parties :

1. – recherche du tri à faire, selon quelle rubrique (4000-4100) ;

2. – lecture séquentielle du fichier avec mise en mémoire (dans le tableau RUB$) des valeurs de la rubrique concernée ; on initialise en outre le tableau de pointeurs IP (4110-4150) ;

3. – classement par échange (méthode du tri à bulle, voir *La Découverte de l'Amstrad*, p. 198 – la différence est qu'ici, ce sont les pointeurs qu'on échange et non les éléments) (4160-4220 et sous-programme d'échange 17000) ;

4. – écriture selon le classement : il suffit de reprendre les pointeurs qui maintenant sont dans l'ordre du classement. C'est là qu'intervient l'accès direct puisque les enregistrements sont listés dans un autre ordre que l'ordre naturel du fichier (4230-4280).

```
3999 REM
4000 REM TRI
4001 REM
4010 INPUT "Nom du Fichier";NOM$ : NOM$=LEFT$(NO
M$,6)
4020 INPUT "Sortie sur Imprimante - oui/non";X$
: IF X$="oui" OR X$="OUI" THEN IMP=8 ELSE IMP=0
4030 GOSUB 11000 ' Lecture Fichier Resume
4040 INPUT "TRI selon quelle rubrique";RU$
4050 INPUT "Croissant (0) ou Decroissant (1) ";T
RI
4060 DIM IP(NE),RUB$(NE)
4070 FOR RR=1 TO NR
4080 IF LIB$(RR)=RU$ GOTO 4110
4090 NEXT
4100 PRINT "Rubrique ";RU$;" non trouvee" : END

4110 FP=0 : FOR E=1 TO NE
4120 GOSUB 12000 ' Lecture d'un Bloc
4130 IP(E)=E
4140 RUB$(E)=MID$(ENR$(EE),D(RR),TA(RR))
4150 NEXT
4160 ECH=0
4170 FOR I=1 TO NE-1
4180 I1=IP(I) : I2=IP(I+1)
4190 IF TRI=0 THEN IF RUB$(I1)>RUB$(I2) THEN GOS
UB 17000 ' Echange
4200 IF TRI=1 THEN IF RUB$(I1)<RUB$(I2) THEN GOS
UB 17000 ' Echange
4210 NEXT
4220 IF ECH=1 GOTO 4160
4230 FOR I=1 TO NE
4240 E=IP(I)
4250 GOSUB 12000 ' Lecture d'un bloc
4260 GOSUB 16000 ' Sortie enregistrement
4270 NEXT
4280 END
```

```
16999 REM
17000 REM ECHANGE POUR LE TRI
17001 REM
17010 ECH=1
17020 IP(I)=I2 : IP(I+1)=I1
17030 RETURN
```

Nous vous conseillons d'ajouter, sur le même modèle, des sous-programmes de recherche sur conditions composées ou de cumul sélectif. Vous avez maintenant toutes les connaissances qu'il faut pour le faire et cette tâche ne devrait pas constituer un exercice trop difficile. Ceci fait, vous disposerez d'un système de gestion de fichiers élaboré, général (applicable dans une foule de domaines) et efficace.

Exercice 3.5. – Le système tel qu'il est permet-il d'implanter un agenda électronique ?

Exercice 3.6. – Le système permet-il de gérer votre compte-chèques ?

FICHIERS INDEXÉS, FILTRÉS

Pour effectuer notre tri, nous avons dû faire un parcours séquentiel du fichier global pour former le tableau de la rubrique argument du classement en mémoire. Ce tableau pourrait être sauvegardé sur disque pour former un nouveau fichier associé ; il pourrait avoir le nom NOM$+"Rn" si n est le n° de rubrique concernée. Il serait d'ailleurs judicieux de mettre en tête de ce fichier le nombre d'enregistrements ; vous comprendrez pourquoi à propos des filtres.

Le classement serait alors simplifié : il suffirait de lire en mémoire le fichier rubrique au lieu de lire le fichier global. Il est judicieux de constituer de tels fichiers rubriques pour toutes les rubriques (il y en a peu de toutes façons) susceptibles de former des arguments de classement ou de sélection. Lorsque ces fichiers rubriques sont associés sur disque au fichier global, on dit que celui-ci est *indexé*. Toutes les opérations de recherche ou classement sont facilitées.

Maintenant, on peut aussi conserver sur disque le tableau de pointeurs IP une fois le classement fait. De même, lors d'une sélection selon une certaine condition, on peut former un tableau IP des n^{os} des enregistrements sélectionnés. Si l'on sauvegarde sur disque sur un fichier appelé NOM$+"Fx", le nombre NS d'éléments sélectionnés et le tableau IP, on dit qu'on a un fichier *filtré* au moyen du filtre x. L'accès aux éléments sélectionnés est immédiat à l'aide du tableau IP. Mais il y a mieux : on peut *composer* deux filtres, c'est-à-dire filtrer avec une deuxième condition un fichier déjà filtré selon une première condition ; on obtient alors les éléments qui satisfont à la fois aux deux

conditions. On peut aussi effectuer un classement sur un fichier filtré. On exploite ainsi au maximum les possibilités de l'accès direct.

Exercice 3.7. – Supposant présent en mémoire le tableau IP de NS éléments correspondant à une certaine sélection, faire le cumul de la rubrique R pour les éléments sélectionnés.

Mise à jour

La mise à jour simple est facile. On affiche un enregistrement : si l'utilisateur tape **RETURN** ou **ENTER**, l'enregistrement est conservé ; si l'utilisateur tape M, on acquiert les nouvelles données et on les introduit à la place des anciennes dans ENR$(EE).

Le programme ci-dessous le fait. Il exploite les sous-programmes déjà existants. Le caractère qui décide de la modification est le caractère que l'on tape en 16520 pour continuer. On donne un numéro d'enregistrement de départ car il est inutile de faire défiler les premiers enregistrements. Lorsqu'on veut terminer, on fait **ESCAPE** ou l'on demande une modification supplémentaire et l'on entre $$$.

```
4999 REM
5000 REM MISE A JOUR
5001 REM
5010 INPUT "Nom du Fichier";NOM$ : NOM$=LEFT$(NO
M$,6)
5020 IMP=0
5030 GOSUB 11000 ' Lecture Fichier Resume
5040 INPUT "Numero de depart";ED
5050 FP=0 : FOR E=ED TO NE
5060 GOSUB 12000 ' Lecture d'un bloc
5070 GOSUB 16000 ' Sortie enregistrement
5080 IF A$=CHR$(13) GOTO 5120
5090 GOSUB 13000 ' Acquisition Enregistrement
5100 GOSUB 14000 ' Ecriture Bloc
5110 IF LEFT$(R$,3)="$$$" GOTO 5130
5120 NEXT
5130 END
```

Mise à jour complète – insertion, suppression

Dans la mise à jour complète, un enregistrement peut être modifié, mais aussi supprimé ou inséré. Dans ce cas, il faut considérer deux fichiers : l'ancien et le nouveau. On lit chaque enregistrement de l'ancien fichier et l'on

attend une lettre de commande par laquelle l'utilisateur dit ce qu'il veut en faire. Si c'est 'm', il y a modification : on acquiert au clavier le nouvel enregistrement et l'on passe l'ancien enregistrement sur l'ancien fichier. Si c'est 'i', il y a insertion : on acquiert le nouvel enregistrement et l'on reste sur place dans l'ancien fichier. Si c'est s, l'enregistrement est supprimé : on avance dans l'ancien fichier mais pas dans le nouveau. Si c'est un autre caractère, l'enregistrement est recopié de l'ancien vers le nouveau fichier.

Comment gérer deux fichiers simultanément alors que l'Amstrad semble l'interdire ? C'est possible avec notre technique de fichiers partiels. Il faut cette fois deux tableaux ENR$ et ENRA$ respectivement pour le nouveau et l'ancien fichiers. De même, la plupart des variables stratégiques sont dédoublées, le nom de celle qui correspond à l'ancien fichier se terminant par un A, par exemple, EA, EEA, etc.

Le programme ci-dessous utilise cette technique. Dans une première phase (de 6000 à 6070), les fichiers partiels sont renommés pour avoir l'extension '.a', tandis que les fichiers nouveaux garderont le nom primitif. Les fichiers anciens sont effacés en fin de programme. Ces opérations font quelquefois imprimer des messages d'erreur ; ne vous en préoccupez pas.

Pour terminer, il ne faut pas oublier de recopier la fin du fichier : l'utilisateur peut soit passer en revue les enregistrements de la fin du fichier ancien et dire qu'ils sont à conserver, ou demander une modification fictive pour l'enregistrement auquel il s'adresse et entrer '$$$'.

Les deux sous-programmes qui s'introduisent sont 12500, copie de 12000 (lecture de bloc), mais pour le fichier ancien et 16200 qui imprime un enregistrement du fichier ancien ; il est très analogue à 16000, sauf que sa fin est adaptée et qu'il n'écrit que sur écran.

```
5999 REM
6000 REM MISE A JOUR COMPLETE
6001 REM
6010 INPUT "Nom du fichier";NOM$ : NOM$=LEFT$(NO
M$,6)
6020 GOSUB 11000 ' lecture fichier resume
6030 NF=1+INT(NE/NG)
6040 FOR F=1 TO NF
6050 F$=MID$(STR$(F),2)
6060 |REN,NOM$+F$+".a",NOM$+F$
6070 NEXT
6080 DIM ENRA$(NG)
6090 NEA=NE : FFA=0
6100 E=1 : F=1 : F$="1" : EE=1
6110 GOSUB 10500 : NE=0 : R$="    "
6120 FOR EA=1 TO NEA
6130 GOSUB 12500 ' Lecture bloc ancien
6140 IF LEFT$(R$,3)="$$$" GOTO 6190
```

```
6150 GOSUB 16200 ' Impression enregistrement anc
ien
6160 IF A$="i" GOTO 6200
6170 IF A$="m" GOTO 6210
6180 IF A$="s" GOTO 6280
6190 ENR$(EE)=ENRA$(EEA) : GOTO 6230
6200 EA=EA-1
6210 GOSUB 13000 ' Acquisition
6220 IF LEFT$(R$,3)="$$$" GOTO 6190
6230 E=E+1 : NE=NE+1
6240 EE=EE+1 : IF EE<=NG GOTO 6280
6250 GOSUB 14000 ' Ecriture bloc
6260 GOSUB 10500 ' Remplissage $
6270 F=F+1 : F$=MID$(STR$(F),2) : EE=1
6280 NEXT
6290 GOSUB 14000 ' Ecriture bloc
6300 GOSUB 10000 ' Ecriture fichier resume
6310 PRINT NE;" Enregistrements ecrits"
6320 NF=1+INT(NE/NG)
6330 FOR F=1 TO NF
6340 F$=MID$(STR$(F),2)
6350 |ERA,NOM$+F$+".a"
6360 NEXT
6370 END
```

```
12499 REM
12500 REM LECTURE D'UN FICHIER PARTIEL ANCIEN
12501 REM
12510 EEA=1+(EA-1) MOD NG
12520 FA=1+INT((EA-1)/NG)
12530 FA$=MID$(STR$(FA),2)
12540 IF FA=FPA THEN RETURN
12550 FPA=FA
12560 OPENIN NOM$+FA$+".a"
12570 FOR G=1 TO NG
12580 INPUT#9,ENRA$(G) : NEXT
12590 CLOSEIN
12600 RETURN
```

```
16199 REM
16200 REM IMPRIME ENREGISTREMENT ANCIEN
16201 REM
16210 PRINT : PRINT
16220 PRINT "Enregistrement ancien No ";EA;"  ("
;FA;EEA;" )"
16230 PRINT
16240 FOR R=1 TO NR
16250 PRINT LIB$(R),MID$(ENRA$(EEA),D(R),TA(R))
```

```
16260 NEXT
16270 PRINT : PRINT "Appuyez sur I pour inserer
avant cet    enregistrement, sur M pour le modif
ier, sur S pour le supprimer, sur une autre  tou
che pour le garder"
16280 A$=INKEY$ : IF A$="" GOTO 16280
16290 RETURN
```

Exercice 3.8. – Simplifiez les instructions 6320 à 6370.

Note importante pour le 464

Sur 464, les paramètres des commandes AMSDOS (qui commencent par ¦) qui sont fournis sous forme de variables ou d'expressions arithmétiques doivent être précédés du signe @. On écrira par exemple ¦ERA,@NOM$. Vous devez donc modifier en conséquence les programmes listés ci-dessus.

RÉCAPITULATION

Nous nous sommes maintenant dotés d'un système de gestion de fichiers très efficace. Il surmonte les principales limitations du BASIC Amstrad en matière de fichiers puisqu'il permet l'accès direct et la manipulation de plusieurs fichiers simultanément.

Les seules limitations qui lui restent sont celles de la faible capacité et de la lenteur des disquettes 3 pouces.

Pour nous donner encore plus d'idées sur les possibilités des fichiers, le chapitre suivant va nous fournir une brève introduction aux bases de données.

CHAPITRE 4
SYSTÈME D'EXPLOITATION DISQUE ET BASES DE DONNÉES

Avant d'aller plus avant, essayons de faire le point sur ce que nous avons appris. Nous avons appris à demander au système – par des commandes – d'accomplir pour nous certaines actions. Mais ces actions sont plus ou moins élémentaires et elles se regroupent par niveaux d'égale complexité.

Par exemple, lire un secteur donné d'une piste déterminée est plus élémentaire que copier un fichier en bloc. Nous avons appelé *primitives* les actions les plus élémentaires et le SED met à notre disposition tout un ensemble de primitives. Malheureusement, il y a des primitives que l'Amstrad se réserve, en particulier les primitives d'accès direct ; c'est pourquoi nous avons été obligés d'en construire au chapitre précédent.

Pour accomplir une action composée, nous pouvons écrire un petit sous-programme qui appelle plusieurs primitives élémentaires. Lorsque nous appelons ce sous-programme, nous ne voyons pas de différence avec l'appel d'une primitive élémentaire. Donc, pour celui qui l'appelle, notre sous-programme constitue une primitive, mais de niveau plus élevé.

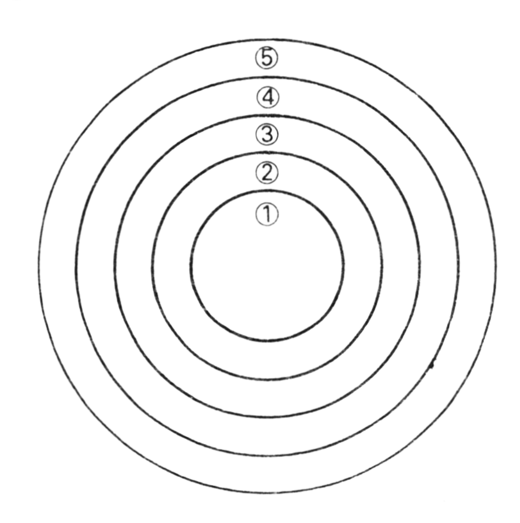
Exemple

Notre programme de mise à jour du chapitre précédent peut être considéré comme réalisant la primitive "correction". Il fait appel à de nombreuses primitives élémentaires.

On a ainsi un emboîtement de niveaux de complexité croissante. A chaque niveau, les actions des niveaux inférieurs sont considérées comme des primitives.

Ainsi, on distingue (*figure ci-dessous*) les primitives élémentaires (1), puis le niveau commandes du SED (2) ; au-dessus, se trouvent le système de gestion de fichiers (3), puis le niveau manipulations de structures de données (4) et enfin le niveau système d'informations ou base de données (5). Les SED AMSDOS et CP/M s'étendent sur les niveaux 1, 2 et 3, encore que AMSDOS soit assez "léger" sur le niveau 3 ; mais CP/M vient prendre le relais.

Il y a un autre classement possible des primitives : par contexte d'utilisation, où l'on considère plusieurs jeux en fonction du contexte. En fait, cette classification peut être considérée comme étant celle des *méthodes d'accès* avec un jeu utilisé en accès direct (que nous avons dû développer au chapitre 3), un jeu utilisé en accès séquentiel et un jeu de primitives de sauvegarde et chargement de programmes.



① Primitives élémentaires
② Commandes simples
③ Système de gestion de fichiers
④ Structures de données
⑤ Bases de données

LES PRIMITIVES DE CP/M

Nous passons brièvement en revue les commandes de CP/M. Celles de AMSDOS ont été vues au chapitre 3. Nous verrons que CP/M Plus est plus riche que CP/M 2.2. N'oublions pas que le jeu de primitives disponibles détermine en dernier ressort la qualité et l'agrément d'un système d'exploitation. CP/M est devenu le standard des systèmes d'exploitation pour les micro-ordinateurs 8 bits.

Nous distinguerons cinq catégories :

1. – commandes portant sur *un* fichier sur disquette,

2. – commandes portant sur une disquette entière,

3. – commandes de transfert d'un périphérique à un autre,

4. – commandes de mode de fonctionnement d'un périphérique,

5. – commandes diverses.

La mention (+) signifie "sur CP/M Plus seul", tandis que (2) signifie "CP/M 2.2 seulement". (A) signifie "a un équivalent en AMSDOS". Nous passons sur certains détails car CP/M Plus est très riche de possibilités.

Commandes portant sur un fichier

ERA nom de fichier (A)

Efface le fichier indiqué. Plusieurs fichiers peuvent être effacés si l'on utilise la dénomination ambiguë, ce qui est dangereux. Notez que dans ces commandes le nom de fichier est fourni sans guillemet.

FILECOPY nom (2)

Copie le fichier indiqué d'une disquette sur une autre dans un système à une seule unité (il y a bien sûr des échanges de disquettes). Sur 6128 utiliser PIP : avec CP/M Plus, si l'on demande l'unité B alors qu'il n'y a qu'une unité, le système imprime un message demandant de changer de disquette.

REN nouveau=ancien (A)

Change le nom d'un fichier.

STAT nom (2)

Affiche l'état du fichier indiqué. STAT tout court affiche la place restante sur la disquette.

STAT nom,$attribut (2)

Impose l'attribut voulu au fichier indiqué. Les attributs possibles sont :

$R/O : lecture seule ;
$R/W : lecture-écriture ;
$SYS : fichier système (n'apparaît pas au catalogue) ;
$DIR : fichier normal (apparaît au catalogue).

Sous CP/M Plus, STAT est remplacée par SET et SHOW.

SHOW (+)

Affiche les paramètres d'une disquette.

SET nom [option] (+)

Impose l'option indiquée au fichier désigné. Les options possibles sont :

[RO] : lecture seule ;
[RW] : lecture-écriture ;
[SYS] : fichier caché (n'apparaît que si l'on fait DIRS) ;
[DIR] : fichier normal ;
[PASSWORD=XXX] : on accède au fichier avec un mot de passe ;
[PROTECT=READ] : fichier protégé.

Si l'on ne fournit pas de nom de fichier, ces options concernent la disquette entière. Voilà une transition vers la section suivante.

Commandes portant sur une disquette entière

DISCKIT3 (+) ou **DISCKIT2** (2 – 6128)

Cette commande regroupe avec un menu les fonctions de formatage, copie totale d'une disquette sur une autre, comparaison de deux disquettes. Ces fonctions ont été vues en détail au début du chapitre 3. Sur 464 ou 664, on utilise les commandes qui suivent.

FORMAT (2 – 464/664)

Formatage d'une disquette. Toute disquette neuve doit subir cette opération.

DISCCOPY (1 unité) ou **COPYDISC** (2 unités) (2 – 464/664)

Copie complète d'une disquette (ou plutôt d'une face) sur une autre.

DISCCHK (1 unité) ou **CHKDISC** (2 unités) (2 – 464/664)

Comparaison de deux disquettes. Vérification de la commande précédente.

DIR (A)

Affiche le catalogue de la disquette. La forme DIR [FULL] (valable seulement sous CP/M Plus) affiche le maximum de renseignements.

Si DIR est suivi d'une désignation de fichier, les renseignements sont fournis pour les fichiers compatibles avec cette désignation.

DIRS ou **DIRSYS** (+)

Affiche les renseignements sur les fichiers système (qui n'apparaissent pas avec DIR).

USER ou **USE n** (A)

Active l'utilisateur numéro n. Commande à éviter.

Les trois commandes suivantes servent à manipuler le CP/M (le changer d'adresse, le réécrire sur disque, etc.). Elles ont peu d'intérêt et ne sont fournies qu'avec CP/M 2.2.

MOVCPM taille* (2)

Crée une version de CP/M de taille (comprise entre 64 et 179 en blocs de 256 octets) réduite.

SYSGEN * (2)

Copie sur la disquette le CP/M qu'on vient de créer par MOVCPM.

BOOTGEN (2)

Copie CP/M plus les variables système.

Commandes de transfert entre périphériques

PIP DESTINATION=SOURCE

Destination et source peuvent être des désignations de fichiers (éventuellement ambiguës) ou des noms de périphérique comme :

A	: première disquette ;
B	: seconde disquette ;
CON	: entrée clavier ou sortie écran ;
AUX	: entrée ou sortie auxiliaire ;
EOF	: marque de fin de fichier (source) ;
LST ou PRN	: imprimante ;
RDR	: entrée RS 232 ;
PUN	: sortie RS 232.

Cette commande est la plus souple de CP/M. Elle permet tous les transferts entre disquettes et entre périphériques, sauf la cassette.

Exemple

Pour lister un fichier sur imprimante :

PIP LST:=ZOZO.XXX

Pour constituer un fichier à partir du clavier :

PIP TOTO.TXT=CON:

TYPE nom de fichier

Fait afficher le contenu du fichier spécifié (il doit être en ASCII). Equivalent de PIP CON:=nom.

CLOAD "nomc" nomd (2)

Copie le fichier cassette nomc sur disquette avec le nom nomd. Notez que nomc est entre guillemets au contraire de nomd.

CSAVE nomd "nomc" (2)

Copie le fichier disque nomd sur cassette avec le nom nomc.

GET nom fichier (+)

Prend le contenu du fichier spécifié comme entrée clavier.

PUT nom fichier (+)

Envoie dans le fichier spécifié ce qui serait sinon envoyé vers l'imprimante.

SETLST nom fichier (+)

Envoie sur l'imprimante le contenu du fichier indiqué. Celui-ci est en principe une suite de caractères de contrôle qui mettent l'imprimante dans un mode convenable. Par exemple, si vous faites :

COPY ETROIT=CON:
^'15'
CTRL Z

SETLST ETROIT mettra l'imprimante en mode condensé.

Ceci introduit la prochaine section où l'on va définir le mode de fonctionnement de divers périphériques.

Commandes de mode de fonctionnement

SET24X80 ON ou **OFF** (+)

Active ou désactive le mode 24 lignes 80 colonnes sur l'écran.

SETKEYS nom fichier (+)

Change les codes des touches du clavier en fonction de données trouvées dans le fichier indiqué. Il y a trois fichiers standard KEYS.CCP (qui convient aux commandes CP/M), KEYS.DRL (adapté à Logo et KEYS.WP (adapté au traitement de textes). Vous pouvez en créer par PIP fichier=CON: et en entrant des lignes du style :

code N S C "signification"

où code est le n° de la touche (voir schéma du clavier), N signifie touche naturelle, S avec SHIFT, C avec CTRL et signification décrit le rôle qu'on veut donner à la touche. On peut avoir N S C ou l'une des trois ou deux d'entre elles, selon qu'on veut que la touche joue le même rôle dans les trois cas ou non.

PALETTE fond texte (+)

Définit les couleurs de fond et de texte. Les codes couleur ne sont pas les mêmes qu'en BASIC (voir le tableau dans le manuel constructeur).

LANGUAGE n (+)

Sélectionne le jeu de caractères correspondant au pays n (USA=0, France=1, ...Espagne=7).

SETSIO (+)

Sans paramètre, on affiche les options du RS 232.
Avec paramètres, on fixe ces options. Les options possibles sont :

RX vitesse	:	vitesse de réception (*ex.* 1200 pour 1200 bauds) ;
TX vitesse	:	vitesse de transmission ;
PARITY NONE	:	pas de parité (autres : EVEN, ODD) ;
STOP 1	:	1 bit d'arrêt (autre valeur possible 2) ;
BITS 8	:	8 bits par mot (va de 5 à 8) ;
HANDSHAKE ON ou OFF	:	active ou désactive la synchro ;
XOFF OFF ou ON	:	active ou désactive le protocole d'échange.

DEVICE (+)

Sans paramètre, on affiche les attributs des périphériques.

Avec paramètres, on les impose. On peut par exemple fixer le débit du RS 232 par DEVICE SIO[1200] ou dérouter les données de l'imprimante vers le RS 232 par DEVICE LST:=SIO.

SETUP (2)

Comme les commandes précédentes, cet utilitaire redéfinit les caractéristiques des périphériques. Il a l'avantage d'être piloté par un menu. On peut en particulier imposer une série de commandes qui seront effectuées lors de tout démarrage ; on peut changer le message d'interrogation, changer les codes clavier, etc.

Commandes diverses

Tout fichier nom.COM est un programme qui s'exécute comme une commande en tapant simplement son nom (sans .COM). Vous pouvez créer les vôtres. Sinon, vous trouverez sur les disquettes :

AMSDOS

Quitte CP/M pour revenir à AMSDOS et à BASIC.

HELP (+)

Affiche des messages d'aide et des précisions sur l'emploi des commandes.

SUBMIT nom fichier

Considère le fichier indiqué comme contenant une suite de commandes CP/M et les exécute à la suite. Vous pouvez constituer le fichier par PIP fichier=CON:

GENGRAF nom (+)

Munit le programme **nom** d'un chargeur des extensions graphiques **GSX**.

ASM

Assembleur 8080.

DDT

Programme d'aide à la mise au point de programmes en langage machine.

DUMP

Affiche le contenu d'un fichier en hexadécimal.

ED

Un éditeur de textes simplifié.

Nous avons maintenant fini notre tour rapide des commandes CP/M. Certaines sont suffisamment élaborées pour qu'on ne les qualifie pas de primitives, et pourtant leur appel est aussi simple que l'appel d'une primitive.

LES UTILITAIRES

Un niveau d'opérations souvent fournies dans les SED des gros ordinateurs est celui des *utilitaires.* On désigne par ce terme toutes les opérations typiques, déjà parfois assez élaborées, qu'il faut sans cesse accomplir pour gérer des données.

Par exemple, l'opération de copie d'un fichier est un utilitaire.

Les opérations qui se présentent le plus souvent sont les suivantes.

Création

Un fichier est construit soit par écriture séquentielle une fois pour toutes, soit par ajouts successifs.

Dans ce dernier cas, il faut disposer d'un utilitaire qui crée le fichier vide au départ.

Dans le cas de l'accès direct, cet utilitaire peut faire les réservations de place et constituer le fichier résumé associé, avec, comme nous l'avons fait au chapitre 3, le répertoire des libellés.

Ajout

Cette opération consiste à ajouter un enregistrement au fichier. Selon le type de fichier manipulé et le périphérique utilisé, cette opération est plus ou moins simple.

L'ajout d'un enregistrement à la fin du fichier est facile ; il est plus difficile d'intercaler un enregistrement au milieu du fichier.

Sur cassette, il faudrait deux magnétophones l'un en lecture, l'autre en écriture et, surtout, pouvoir ouvrir deux fichiers simultanément.

Au chapitre 3, nous avons procédé par recopie. Il nous a fallu user d'expédients pour simuler l'ouverture simultanée de deux fichiers, mais cette limitation est propre à l'Amstrad.

Nous verrons que si les données sont organisées en liste, l'opération se simplifie car l'on effectue l'ajout à la fin du fichier.

Mise à jour

Cette opération consiste à modifier un ou plusieurs enregistrements. En accès direct, l'opération est très simple. En accès séquentiel, il faut procéder par recopie.

Exercice 4.1. – Dresser un ordinogramme général de la mise à jour d'un enregistrement en accès séquentiel.

Suppression

Cette opération consiste à ôter un enregistrement du fichier. Par exemple, dans le fichier des impayés, il faut supprimer un client qui vient de payer. En accès séquentiel, on procède par recopie. En accès direct, on pourrait se ramener à une mise à jour simple : on pourrait réserver dans chaque enregistrement une zone dite de validité ou d'activité, pouvant prendre deux valeurs : l'une signifiant "enregistrement valide", l'autre signifiant "enregistrement supprimé".

Dès lors, la suppression consiste simplement à changer la valeur de cette zone dans l'enregistrement concerné.

Ce n'est que lorsqu'il y a beaucoup de "trous" dans le fichier qu'un utilitaire dit de "recollage des morceaux", en anglais "garbage collection", réécrit le fichier en supprimant physiquement les enregistrements inactifs.

Exercice 4.2. – Dessinez un ordinogramme général de "garbage collection".

Dans le système de fichiers du chapitre 3, dans le cas d'un fichier *filtré* (dont on possède le tableau IP), on pourrait effectuer la suppression d'un enregis-

trement en supprimant l'élément correspondant du tableau IP et non l'enregistrement lui-même. De temps en temps, il faudrait faire un garbage collection.

Fusion

Cette opération consiste à réunir deux ou plusieurs fichiers en un seul. La version la plus simple est celle où le fichier résultant est formé par la mise bout à bout des deux fichiers de départ.

Plus complexe est la fusion avec *interclassement* : les deux (ou plus) fichiers séquentiels de départ sont classés suivant un certain critère qui sera maintenu dans le fichier final.

L'algorithme est le suivant : à tout moment réside en mémoire un enregistrement provenant de chacun des fichiers de départ. L'enregistrement à écrire sur le fichier final est déterminé par comparaison. Ensuite, un nouvel enregistrement est lu sur le fichier d'où provenait l'enregistrement qui vient d'être écrit.

Tri

Cette opération consiste à produire à partir d'un fichier non classé, un fichier classé suivant un certain critère (par exemple : clients par ordre alphabétique, représentants par ordre de chiffre d'affaires réalisé dans l'année, élèves par notes décroissantes, etc.).

Comme on l'a vu au chapitre précédent, le tri est facile en accès direct : il faut commencer par une lecture séquentielle dans chaque enregistrement. Former ainsi un tableau en mémoire. Ensuite, par des opérations mémoire, un tableau de pointeurs conforme au classement est constitué. Il suffit alors de lister ou de copier le fichier en prenant les enregistrements dans l'ordre indiqué par le tableau de pointeurs, ce qui est possible en accès direct.

En accès séquentiel, et en faisant abstraction de la limitation de l'Amstrad sur l'ouverture simultanée de plusieurs fichiers, il est possible d'appliquer la méthode du tri à bulle (*cf. La découverte de l'Amstrad*, p. 198). A chaque passe, le fichier initial est remplacé par un fichier mieux classé ; à chaque passe, tout le fichier est parcouru en comparant à chaque niveau deux enregistrements consécutifs ; s'ils sont dans le bon ordre, ils sont recopiés tels quels ; s'ils sont dans le mauvais ordre, ils sont intervertis. Le tri est terminé lorsqu'une passe a été réalisée sans inversion. Cette méthode fait lire tout le fichier, un nombre interminable de fois.

Une meilleure méthode consiste à lire une partie du fichier en mémoire (un segment), la plus longue possible qui tienne dans la mémoire. Ensuite cette partie est classée par des opérations en mémoire et puis copiée – classée – sur un fichier intermédiaire. Puis l'opération est recommencée pour le segment suivant. Si le fichier est dix fois plus grand que la mémoire disponible, il y aura dix segments donc dix fichiers intermédiaires. Le tri se termine par une opération d'interclassement.

Toutes ces opérations utilitaires sont définies quel que soit le fichier mais elles se réalisent différemment (plus ou moins facilement) selon la méthode d'accès et la structure ou l'organisation des données. La plupart des opérations sont possibles en séquentiel, à condition de pouvoir manipuler plusieurs fichiers simultanément : l'accès direct n'est pas toujours nécessaire. Voyons maintenant comment organiser nos données.

STRUCTURES DE DONNÉES

Jusqu'ici, la seule organisation que nous avons envisagée pour nos données a consisté à les juxtaposer les unes à côté des autres, ou, si elles étaient coordonnées, les unes après les autres, un peu comme les éléments d'un tableau BASIC.

Cela s'appelle l'organisation en file, ou encore l'organisation séquentielle (même si le fichier est en accès direct, ce n'est pas le même problème).

Mais il y a d'autres organisations de données, qui peuvent se révéler plus utiles dans certaines applications. Passons très brièvement en revue les plus fondamentales d'entre elles.

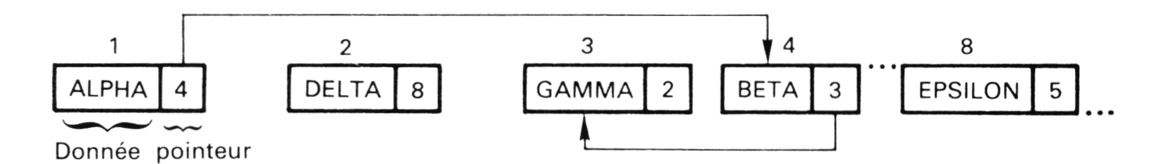
Liste (ou chaînage)

Lorsque les données sont organisées en liste, les enregistrements ne sont pas disposés dans l'ordre mais chaque enregistrement contient deux zones : la zone donnée proprement dite et la zone pointeur.

Le pointeur indique l'adresse (par exemple sous forme de numéro d'enregistrement) de l'enregistrement qui, conformément à l'ordre considéré, est le successeur de l'enregistrement concerné.

Exemple

Etant donné l'ordre ALPHA, BETA, GAMMA... voici une structure de liste :



Il faut disposer d'un pointeur spécial qui détermine le premier enregistrement, ou alors, souvent, la convention est d'assurer que le premier enregistrement sera aussi le n° 1.

L'utilisation du pointeur pour retrouver l'enregistrement successeur n'est efficace que si la structure est implantée sur un fichier en accès direct.

Le dernier enregistrement de la liste possède un pointeur qui a une valeur spéciale pour indiquer qu'il est le dernier, par exemple 0 ou FF en hexadécimal.

Exemple



Si le dernier enregistrement pointe vers le premier (dans l'exemple ci-dessus, le pointeur serait égal à 1), la liste est dite *circulaire.*

Liste réciproque

On peut aussi gérer un autre jeu de pointeurs où chaque enregistrement pointe vers son prédécesseur ; la liste est dite *réciproque.*

Lorsque les deux jeux de pointeurs sont gérés simultanément, la liste est dite *double.*

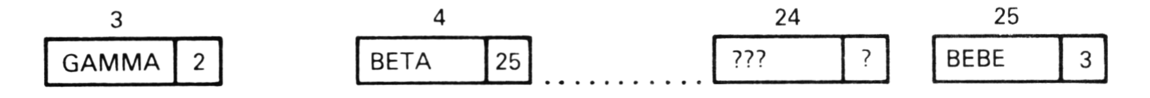
Lorsque plusieurs jeux de pointeurs sont gérés, conformément à plusieurs critères différents, la liste est dite *multiple.*

Mais il ne faudrait pas avoir une quantité de pointeurs telle que la partie pointeurs devienne plus encombrante que les informations proprement dites !

Une grande partie des opérations utilitaires se simplifie si les données sont organisées en liste. Par exemple, pour lister les éléments dans l'ordre, il suffit de suivre les pointeurs. Une suppression reviendra à modifier le pointeur du prédécesseur de l'enregistrement à supprimer afin qu'il pointe vers le successeur.

Exercice 4.3. – Dans l'exemple ci-dessus, supprimez l'enregistrement EPSILON.

Un ajout se fera de la façon suivante : l'enregistrement à intercaler se met physiquement à la fin du fichier et les pointeurs sont modifiés. Par exemple, pour insérer BEBE entre BETA et GAMMA :



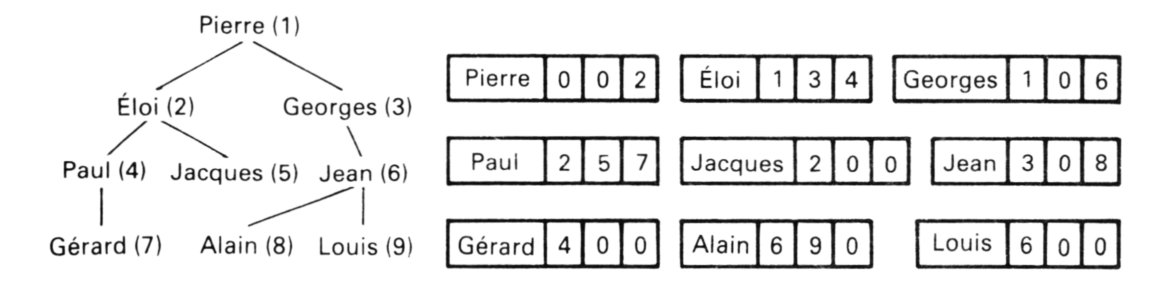
Arbres (ou arborescences)

Cette structure est bien connue : c'est celle des arbres généalogiques. Le premier élément s'appelle la racine, les éléments terminaux s'appellent les feuilles.

Une manière possible de représenter un arbre est la suivante (nous ne prétendons pas que c'est la meilleure) : l'ensemble des "fils" d'un même "père" forme une liste, où chaque "fils" a un pointeur vers le "frère" qui le suit. Chaque père a un pointeur vers son premier fils. Les fils de la dernière génération (feuilles) ont 0. Chaque fils a un pointeur vers son père. Le "patriarche" (racine) a 0.

Exemple

Entre parenthèses figure le n° d'enregistrement ; les pointeurs sont dans l'ordre père, frère, fils :

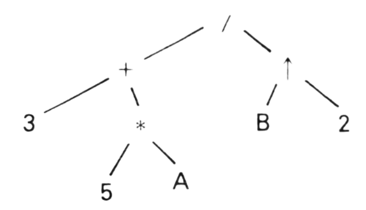


Un arbre dans lequel tous les pères ont deux fils s'appelle un *arbre binaire*.

Nous n'insisterons pas plus sur les arbres mais nous noterons que cette structure intervient dans beaucoup de problèmes informatiques.

C'est le cas de tous les problèmes où l'on doit représenter une hiérarchie comme ensemble des utilisateurs d'un centre de calcul, un utilisateur, les fichiers de cet utilisateur.

Les arbres interviennent en analyse syntaxique (quand BASIC cherche si une de vos instructions est correcte) ou pour interpréter une expression arithmétique. Par exemple, compte tenu des règles d'évaluation, (3+5*A)/B^2 correspond à l'arborescence :



Hachage (en anglais hash-coding)

Supposons un fichier clients organisé autour du numéro de chaque enregistrement. Si le numéro a 5 chiffres, cela permet 99999 clients ! Il n'y en aura jamais autant ! On peut tenir pour vraisemblable qu'il n'y aura pas plus de, par exemple, 1000 clients et par suite réserver la place pour 1000 enregistrements.

On appelle *fonction de hachage* une fonction mathématique qui assigne un numéro d'enregistrement (parmi 1000) en fonction du numéro de client.

Cette fonction est déterminée mathématiquement (nous ne nous étendrons pas sur les techniques effectivement employées) pour que la répartition des enregistrements soit le plus uniforme possible sur le fichier : la fonction de hachage ressemble donc un peu à la fonction RND qui permet à BASIC de calculer des séquences pseudo-aléatoires.

Une fois donnée la fonction de hachage, il est possible d'écrire un nouvel enregistrement de numéro client donné, ou de retrouver tout client de numéro donné. Cela suppose, bien sûr, l'accès direct.

Il n'y a qu'un problème : parfois deux numéros de client distincts peuvent conduire au même numéro d'enregistrement ; il y a, comme on dit, *collision*. Ce problème est résolu en adjoignant au fichier une zone organisée en liste (appelée zone de débordement) dans laquelle sont rangés les enregistre-

ments "en double" (appelés *synonymes*). L'enregistrement placé à l'adresse calculée par hachage contient un pointeur vers ses synonymes.

Si la taille prévue pour le fichier et la fonction de hachage sont bien adaptées, il ne doit pas y avoir beaucoup de collisions.

BASES DE DONNÉES

Une base de données est constituée lorsque :

1. – Un vaste ensemble de données utiles à une application (ou à un groupe d'applications déterminées) est regroupé en un ou, plus souvent, plusieurs fichiers.

2. – Les programmes de traitement de ces données dans le cadre de l'application considérée ont été réunis.

Ceci suppose que :

3. – Les relations et les interconnexions qui lient ces données entre elles ont été examinées et explicitées. Par exemple, un bon de commande est relatif à un client ; il doit donc en rappeler le numéro.

4. – Compte tenu de ces relations et des traitements que doivent subir les données, celles-ci ont été réparties en fichiers et les structures d'organisation ont été choisies. Les structures d'organisation peuvent être différentes pour chacun des fichiers. Il faut prévoir convenablement la taille des fichiers.

Constituer une base de données revient à modéliser l'entreprise. Il faut essayer de rester simple, ne pas chercher à obtenir un système qui fasse tout car il serait inextricable.

Trop de mécomptes ont été enregistrés, dont, peut-être, la modestie des micro-ordinateurs saura nous préserver.

Transactions et événements

Le fonctionnement de la base de données doit suivre la vie de l'entreprise. Celle-ci est marquée par toutes sortes d'événements qu'il faut recenser, par exemple l'arrivée d'un bon de commande.

Un événement de base entraîne d'autres événements qui se traduisent en actions sur la base de données.

Par exemple, l'arrivée d'un bon de commande va entraîner la préparation d'une facture en attente : elle va entraîner une mise à jour des stocks des produits qui entrent dans la fabrication des articles commandés, ce qui pour un produit arrivant à épuisement, entraîne l'émission d'une commande. Il y aura en outre une mise à jour du chiffre d'affaires réalisé avec le client concerné dans le fichier clients et une mise à jour du chiffre d'affaires réalisé par le représentant qui s'occupe de ce client dans le fichier représentants. Si la comptabilité est également traitée, il y aura un ajout dans le fichier "journal des ventes".

Le plus souvent, les interventions sur la base de données se font sous la forme de transactions conversationnelles, avec dialogue entre un employé et le clavier-écran.

Certaines transactions sont simples et se ramènent à la consultation ou l'interrogation d'un seul fichier de la base comme, par exemple, savoir si le client "x" est à jour de ses paiements.

D'autres transactions sont plus complexes et entraînent des événements secondaires comme c'est le cas, par exemple, de l'enregistrement d'un bon de commande.

Les programmes sont écrits de façon à permettre un dialogue homme-machine le plus commode possible.

Par exemple, le programme demande à l'utilisateur quelle transaction il souhaite accomplir en affichant un "menu" sur l'écran :

```
[] CONSULTATION FICHIER CLIENTS
[] CONSULTATION FICHIER REPRESENTANTS
[] CONSULTATION JOURNAL DES VENTES
[] ENREGISTREMENT COMMANDE
[] ENREGISTREMENT PAIEMENT
[] EMISSION FACTURE
etc.
```

La fonction désirée est sélectionnée en amenant le curseur sur la ligne correspondante. Une fois le choix fait, pour une fonction complexe, un *sous-menu* peut être affiché, et ainsi de suite...

Pour une fonction d'acquisition de données, il faut que la machine interroge l'utilisateur, et s'il y a des recoupements possibles, vérifie les données introduites et proteste en cas d'erreur. Ainsi, pour enregistrer un nouveau client :

```
No. de Client ? 19A85
***PAS DE LETTRE DANS LE NUMERO***
No. de Client ? 19185
Nom ? DUPONT
etc...
```

Le langage utilisé doit être le plus proche possible de la *langue naturelle*. Ce n'est pas totalement possible surtout sur les micro-ordinateurs.

Mais pouvoir demander une recherche sous la forme :

NOM=DUPONT

plutôt que sous la forme :

numéro de rubrique ,DUPONT

accomplit un pas dans ce sens. Nous l'avons fait au chapitre 3.

En résumé, nous nous rappellerons que le plus important dans une base de données est de **recenser les informations** nécessaires pour les applications à réaliser. Il serait très ennuyeux d'avoir à ajouter une zone oubliée à tous les enregistrements d'un fichier.

En somme, cela revient à dessiner le type-article des différents fichiers compte tenu des relations qui existent entre les informations. Il faut aussi *prévoir la taille des fichiers* avec soin : selon l'ordre de grandeur de cette taille, des techniques différentes (hachage, liste) peuvent être mises en œuvre.

Nous terminons par de brefs conseils sur l'entretien de votre "discothèque". Ils complètent ceux déjà donnés au début du chapitre 3.

1. – Toute disquette doit être étiquetée, avoir un numéro unique, et un nom.

2. – Tenez un répertoire de vos disquettes, indiquant le contenu de chacune d'elles.

3. – Faites imprimer le répertoire de chaque disquette à l'imprimante et veillez à ce que ce répertoire accompagne toujours la disquette.

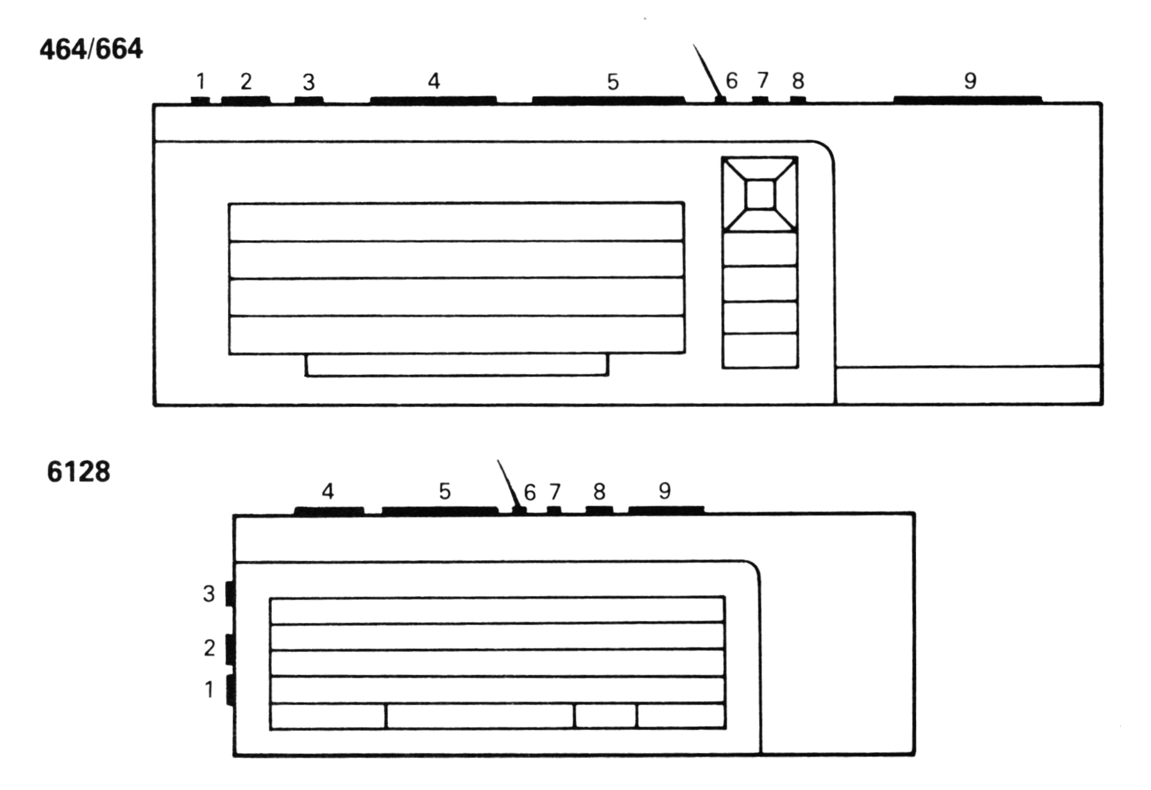
4. – Faites, journellement en usage professionnel, sinon, assez souvent, une ou plusieurs copies de sécurité de chaque disquette.

5. – En usage, P.M.E. gardez une trace imprimée de toutes les transactions de la journée. Si une disquette se trouve abîmée, la copie de sécurité de la veille permettra, en refaisant toutes les transactions de la journée, de reconstituer un fichier à jour.

CHAPITRE 5
PÉRIPHÉRIQUES DIVERS

VUE GÉNÉRALE DES CONNEXIONS À UN AMSTRAD

Les schémas ci-dessous donnent une idée de ce qu'on peut connecter à un Amstrad.



1. – **Prise stéréo** : vous pouvez relier cette prise à votre chaîne hi-fi pour obtenir des effets sonores spectaculaires. Un simple couple de mini-enceintes pour walkman donne déjà de bons résultats.

2. – **Connecteur joystick** : on y relie principalement une manette de jeu (manche à balai). Voir la fonction JOY dans *La découverte de l'Amstrad*, p. 158. On peut aussi y connecter le *stylo lumineux* Amstrad que nous étudions plus loin dans ce chapitre.

Brochage



(Vue arrière)

Le stylo lumineux se relie aux broches 2 et 8.

3. – **Connecteur magnétophone** (664/6128 seulement).

4. – **Connecteur imprimante** : c'est le connecteur pour interface Centronics. Attention au fait qu'il est en 7 bits de données (le bit 7 est toujours à 0). Outre des imprimantes, il existe des traceurs de courbes qui obéissent à la norme Centronics ; vous pouvez donc envisager de les utiliser avec votre Amstrad.

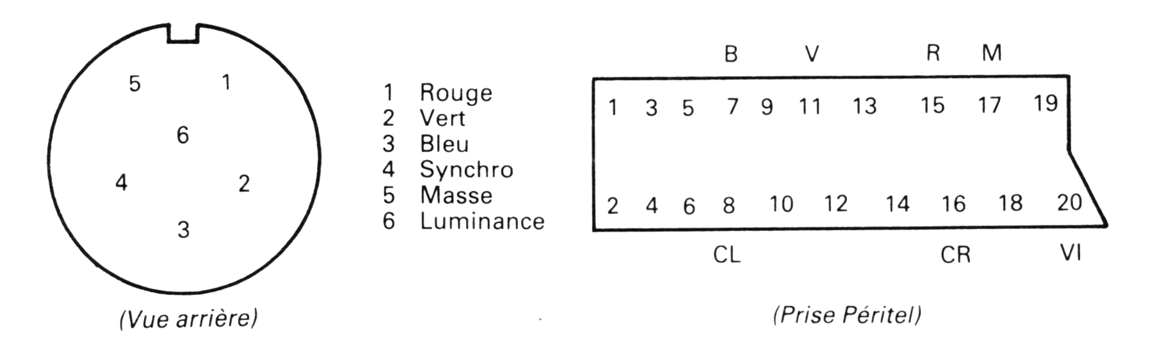
5 – **Connecteur d'extensions** : c'est le connecteur principal de l'Amstrad ; il transmet les signaux de bus du microprocesseur. Sur 464, on y connecte la première unité de disques. Sur tous les modèles, on y connecte diverses extensions comportant une mémoire morte auxiliaire. On y connecte l'interface RS 232 (étudiée dans ce chapitre) ; une unité de synthèse vocale est annoncée.

6. – **Prise 12 V** : se relie au moniteur.

7. – **Prise 5 V** : se relie au moniteur.

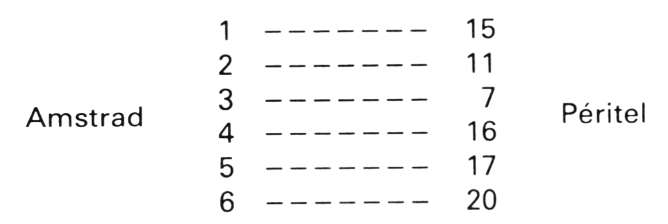
8. – **Prise vidéo** : elle se relie aussi au moniteur, bien sûr. Nous donnons ci-dessous le brochage et un schéma de cordon Péritel qui vous permet d'utiliser un téléviseur couleur ordinaire. Cela marche même avec un téléviseur SECAM et cela peut vous économiser quelque argent : vous pouvez acheter votre Amstrad avec moniteur noir et blanc et avoir tout de même une image couleur. Dans cette hypothèse, vous alimentez

l'Amstrad avec le moniteur noir et blanc. Attention, sur le téléviseur, l'affichage 80 colonnes n'est bien lisible que si le téléviseur est de qualité.



(Vue arrière) (Prise Péritel)

Cordon Amstrad-Péritel



Amstrad	Péritel
1	15
2	11
3	7
4	16
5	17
6	20

Avec certains téléviseurs, il faut en plus alimenter la broche 8 de la prise Péritel avec du 9 à 12 V : une pile 9 V fait très bien l'affaire.

9. – **Connecteur disquette supplémentaire** : ce connecteur est absent du 464 ; il se trouve dans ce cas sur la première disquette.

L'INTERFACE RS 232

La possibilité d'utiliser cette interface sur l'Amstrad est très intéressante. Il s'agit en effet d'une interface standard immensément répandue dans le monde des ordinateurs. De nombreux périphériques sur le marché en sont munis : la présence de cette interface, et surtout le fait que le boîtier contienne une ROM qui renferme le logiciel qui la gère en ajoutant quelques instructions à BASIC, peut vous permettre de connecter de tels périphériques à votre Amstrad. Parmi ces périphériques, il y a notamment des imprimantes et des traceurs de courbes. Mais surtout, l'interface RS 232 est l'interface standard

des télécommunications entre ordinateurs. Elle vous permettra de relier votre Amstrad à des réseaux sur lesquels vous pourrez interroger des bases de données.

QUELQUES CARACTÉRISTIQUES DE LA NORME RS 232

Nous ne donnons pas trop de détails sur les communications série, renvoyant pour cela à D.-J. David, *Les systèmes à microprocesseur*, Editests. La caractéristique fondamentale des communications RS 232 est qu'elles sont en série, c'est-à-dire qu'un octet n'est pas transmis d'un coup mais chaque bit est envoyé après l'autre.

Cela procure une économie sur le nombre de fils : au lieu de 8 fils pour transmettre tous les bits en même temps, on n'a plus besoin que d'un fil sur lesquel les différents bits circulent successivement. De fait, il y a un mode de fonctionnement du RS 232 où il n'y a que trois fils, la masse, donnée émise et donnée reçue.

La norme RS 232 fixe deux choses :

– la forme du connecteur standard RS 232 qui est une prise "Cannon" 25 broches et la liste des signaux ainsi que le brochage ;

– les propriétés électriques : lorsque le signal RS 232 est à 0, la tension doit être comprise entre +3V et 12V ; lorsqu'il est à 1, la tension doit être comprise entre –3V et –12V.

Voici le brochage :

1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25

Broche	Rôle	
2	Sortie donnée émise	(TXD)
3	Entrée donnée reçue	(RXD)
4	Sortie demande émission	(RTS)
5	Entrée prêt à émettre	(CTS)
6	Entrée modem prêt	(DSR)
7	Masse	
8	Entrée détection porteuse	(DCD)
20	Sortie terminal prêt	(DTR)
22	Entrée appel	(RI)

Les autres broches sont inutilisées. Les broches autres que la masse et les deux broches de données servent à des signaux de synchronisation des échanges.

DIFFÉRENTS PARAMÈTRES PROGRAMMABLES

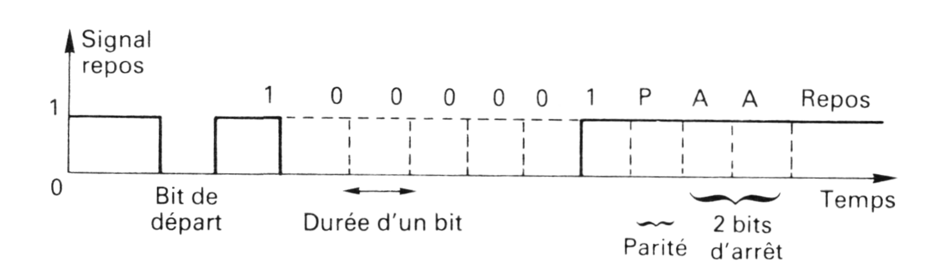
Le problème principal est celui de la synchronisation. Elle est réalisée de deux manières :

1. – on fixe la fréquence des bits ;
2. – on définit le départ de chaque caractère.

Pour cela, lorsqu'une ligne est au repos, elle est au niveau 1. Le départ est marqué par l'émission d'un bit à 0, dit bit de départ. Ensuite, on a les 5 à 8 bits (nombre programmable) qui forment le caractère. Il y a ensuite éventuellement 1 bit de parité, puis 1 ou 2 bits à 1, dits bits d'arrêt. La ligne reste alors à 1, un temps variable jusqu'au prochain caractère. Notez que les bits transmis le sont en commençant par le moins significatif.

Exemple

Soit à transmettre un A en ASCII (code en 7 bits hexadécimal 41, décimal 65, binaire 1000001). Si le bit de parité est à 1 (parité impaire), le signal transmis en fonction du temps sera :



La vitesse se mesure en bauds, unité qui est voisine du bit par seconde. Les vitesses possibles avec l'interface Amstrad vont de 50 à 19200 bauds. La vitesse par défaut est 9600 bauds. Nous avons fait nos essais à 1200 bauds, ce qui est bien assez rapide ; n'oubliez pas que la fiabilité diminue si la vitesse augmente.

PROGRAMMATION DE L'INTERFACE RS 232 DE L'AMSTRAD

La programmation du RS 232 se fait soit par des instructions ajoutées à BASIC (elles commencent par ¦), soit par des commandes CP/M. Nous n'étudions ici qu'un petit nombre des commandes BASIC supplémentaires. Notons

que pour utiliser le RS 232 sous CP/M Plus (6128), il y a une correction à implanter dans le système ; elle est décrite dans la notice du constructeur.

Les deux situations les plus utiles sont la connexion d'une imprimante RS 232 à votre Amstrad et la connexion de votre Amstrad à un autre ordinateur pour échanger des données appartenant à des fichiers. Ce sont celles pour lesquelles nous avons fait des essais effectifs. Mais il y en a d'autres, permises par l'Amstrad, notamment l'émulation d'un terminal PRESTEL (c'est le TELETEL anglais).

Dans tous les cas, la première chose à faire est de fixer les paramètres de la transmission, ce qui se fait par l'instruction |SETSIO.

```
|SETSIO,VE,VR,C,BD,P,BA
```

où VE (50 à 19200) est la vitesse d'émission (seul paramètre obligatoire), VR est la vitesse de réception, C décide si l'on tiendra compte (C=1) ou non (C=0) des signaux de contrôle (CTS, DCD, etc.), P est la parité (0=pas de parité, 1=impaire, 2=paire), BD (5 à 8) est le nombre de bits de données et BA définit le nombre de bits d'arrêt (0=>1, 1=>1,5 et 2=>2 bits d'arrêt).

Connexion d'une imprimante

Nous avons fait des essais avec une Brother EP44. La liaison 3 fils s'est avérée suffisante :

Amstrad		Brother
2	________	3
3	________	2
7	________	7

Cela étant, après un |SETSIO,1200 qui fixe la vitesse à 1200 bauds (par exemple), il suffit de faire :

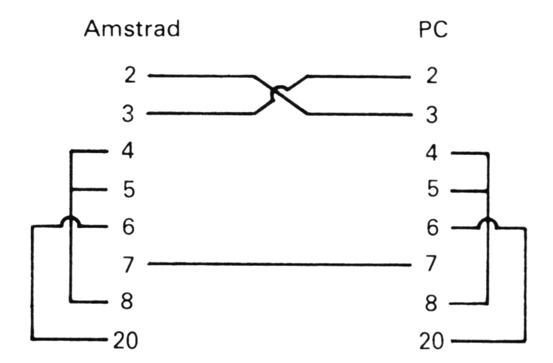
```
|SERIAL
```

qui redirige la sortie imprimante vers le RS 232. On fait ensuite des PRINT#8, ... ou LIST ...,#8.

On annule le |SERIAL par |PARALLEL.

Échange de fichiers

Nous avons fait des essais en connectant l'Amstrad à un compatible PC. Outre les connexions vues ci-dessus, il a fallu connecter de chaque côté du câble les broches 4,5 et 8 d'une part et 6 et 20 d'autre part, d'où le schéma du câble en haut de la page ci-contre.



Cela étant, vous fixez les paramètres de la transmission (surtout la vitesse) par un ¦SETSIO (*N.B.* – Il faut aussi les fixer à des valeurs compatibles du côté de l'autre ordinateur ; avec un compatible PC, cela se fait par une commande MODE).

Pour envoyer un fichier depuis l'Amstrad, il vous faut constituer ce fichier sous forme ASCII, si l'autre ordinateur n'est pas un Amstrad (sinon, il risque de ne pas être "compris" de l'autre côté). Pour un fichier programme, par exemple, vous le chargez, puis vous faites un SAVE "nom",A.

Ensuite, vous utilisez la commande :

¦OUTFILE,"nom"

pendant que du côté du PC, vous aurez exécuté la commande :

COPY COM1: A:nom de fichier

Pour un programme présent en mémoire, vous pouvez aussi faire ¦SERIAL puis LIST#8 et PRINT#8,CHR$(26). Le caractère de code 26 est la marque de fin de fichier envoyée automatiquement par ¦OUTFILE.

Pour recevoir un fichier, vous faites :

¦INFILE,"nom"

pendant que du côté PC, vous exécutez un COPY fichier COM1:.

Bien sûr, si la communication est entre deux Amstrad, on utilise ¦OUTFILE dans l'un et ¦INFILE dans l'autre. Entre deux Amstrad, on peut aussi utiliser le couple :

¦SUCK,"nom" (réception)

et

¦BLOW,"nom" (émission)

qui offre un protocole de récupération des éventuelles erreurs de transmission.

Pour que la transmission aboutisse au clavier-écran de votre Amstrad, vous faites un :

¦TERMINAL

Il y a aussi des instructions plus élémentaires, plus *primitives* au sens du chapitre 4 : le couple ¦INCHAR, ¦OUTCHAR qui, respectivement, émet et reçoit un caractère et le couple ¦INBLOCK, ¦OUTBLOCK qui agit au niveau d'un bloc de caractères.

¦INCHAR (et ¦OUTCHAR, qui a la même) a une syntaxe un peu particulière :

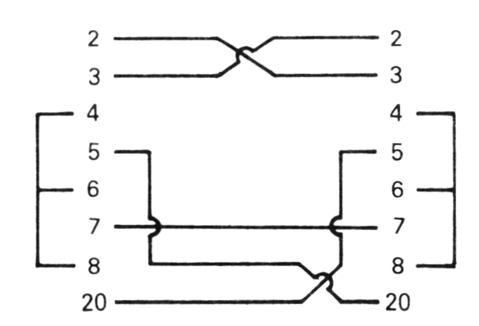
¦INCHAR,@S%,@C%

où S% reçoit un code d'erreur (0 si le caractère a bien été reçu) et C% est une variable entière qui contiendra le code ASCII du caractère reçu. Ce sont les adresses des paramètres que l'on fournit, d'où les @.

Le petit programme ci-dessous reçoit des caractères et il les imprime aussitôt à l'écran.

```
5 |SETSIO,1200
10 s%=0:c%=0
20 |INCHAR,@s%,@c%
30 IF s%<>0 GOTO 20
35 IF c%=26 GOTO 5
40 PRINT CHR$(c%);
50 GOTO 20
```

Attention, pour utiliser des routines élémentaires de cette sorte, le câble de liaison doit être différent de celui vu ci-dessus : il faut assurer une synchronisation des échanges plus fine. Le schéma ci-dessous devrait convenir :



LE STYLO LUMINEUX

Amstrad commercialise un "stylo lumineux". Rappelons que ce périphérique permet de repérer des points sur l'écran afin que, sous contrôle d'un programme convenable, l'ordinateur "connaisse" et exploite les coordonnées des points voulus.

Le stylo lumineux Amstrad se connecte sur le connecteur joystick. Nous avons eu quelques problèmes avec le stylo de pré-série utilisé pour nos essais. Si lorsque vous le connectez, votre écran se couvre de caractères "flèche en bas", faites ESC. Si le problème continue, c'est que le stylo présente une impédance trop faible. Pour notre part, nous avons remplacé la résistance de 2,2 K à l'intérieur par une résistance de 10 K. Il ne faut pas non plus que la résistance soit trop grande, sinon le stylo ne "verra" plus rien. Quoi qu'il en soit, il vaut mieux confier toute intervention à votre revendeur.

D'autre part, le stylo Amstrad ne "sait" repérer sur l'écran que des pavés assez gros (au moins 4×4 pixels). Aussi, il vaut mieux que la couleur du fond soit sombre, que la couleur des pavés soit brillante et qu'il n'y ait pas trop d'éclairage parasite dans la pièce où vous travaillez. Vous pouvez être amené à augmenter le réglage de luminosité sur votre moniteur. Rappelons qu'Amstrad précise que le stylo lumineux ne fonctionne pas avec le moniteur monochrome. Il fonctionne avec un téléviseur relié à l'aide du câble Péritel décrit ci-dessus. Le stylo lumineux ne fonctionne pas si l'interface RS 232 ou le synthétiseur vocal sont installés.

Le stylo Amstrad est livré avec un logiciel sur cassette, que vous pouvez transférer sur disquette. Il est formé d'une routine en langage machine (implantée de 40000 à 41560) et d'un programme BASIC qui contient un mode d'emploi et un excellent programme de dessin sur l'écran.

Nous ne dirons rien sur le programme de dessin, sinon pour signaler qu'il a d'excellentes possibilités, notamment la sauvegarde de votre dessin sur cassette ou disquette, la recopie sur imprimante (DMP 1), la recopie d'une zone d'écran sur une autre, le "zoom", etc. Signalons qu'en mode "dessin à la main", vous devez maintenir la barre d'espace appuyée et surtout ne pas déplacer le stylo trop vite pour que le système puisse vous suivre.

Pour utiliser le stylo dans vos propres programmes, il vous suffit d'étudier avec soin le mode d'emploi fourni par le programme BASIC ainsi que le programme lui-même. Il en ressort que, pour des applications de type menu, vous devez incorporer à votre programme :

1. – la routine en langage machine (et, bien sûr, protéger sa zone mémoire par un MEMORY 40000) ;

2. – le sous-programme BASIC de 30000 à 30130 qui fait partie du logiciel livré.

Ce sous-programme est contrôlé par les paramètres NU, WA, TX, TY et TS. Il va afficher NU pavés, disposés horizontalement si WA=1, verticalement si WA=0. TX et TY sont les coordonnées du premier pavé et TS est l'écart (en mailles caractères) entre pavés. Au retour, on a dans NU le numéro du pavé qui a été sélectionné.

Bien entendu, avant d'appeler 30000, il faut afficher sur l'écran (par des LOCATE et PRINT) les éléments de menu que vous voulez proposer.

Exemple

Nous traitons un exemple très simple. On veut proposer à l'utilisateur de choisir entre OUI et NON, disposés ainsi que l'écran :

		col.	5	7
ligne	10		[]	OUI
	12		[]	NON

On a donc NU=2 (2 choix), WA=0 (vertical), TX=5, TY=10 et TS=2. Il faut imprimer OUI en 10,7 et NON en 12,7. Avec le logiciel livré chargé, vous pouvez ajouter le petit programme suivant :

```
49999 END
50000 CLS
50010 LOCATE 7,10 : PRINT "OUI"
50020 LOCATE 7,12 : PRINT "NON"
50030 NU=2 : WA=0 : TX=5 : TY=10 : TS=2
50040 INK 1,26 : PEN 1 : GOSUB 30000 : LOCATE 5,
20
50050 IF NU=1 THEN PRINT "Vous avez choisi OUI"
ELSE PRINT "Vous avez choisi NON"
```

Vous remarquerez en 50040 que l'on assure que les pavés soient blancs brillants avant l'appel de 30000 ; en effet une couleur plus sombre risque de ne pas être décelée. Après l'appel, un LOCATE repositionne le curseur : le sous-programme le laisse juste en dessous du pavé sélectionné.

Essayez ce petit programme sous les trois modes d'affichage. Vous constaterez qu'il faut insister, en mode 80 colonnes, pour que le pavé soit décelé.

L'utilisation du stylo lumineux devrait maintenant être claire. Si vous en possédez un, nous vous suggérons, à titre d'exercice, de l'utiliser pour choisir une fonction dans le menu du système de fichiers du chapitre 3.

LE MANCHE A BALAI

Nous ne quittons pas tout de suite le connecteur joystick. L'utilisation du manche à balai est très simple grâce à la fonction JOY décrite dans *La découverte de l'Amstrad.* Toutefois, à la demande d'un certain nombre de lecteurs, nous donnons ci-dessous une version joystick du programme *TELECRAN* (p. 203 de *La découverte de l'Amstrad*).

Vous obtenez un tracé si, en même temps que le déplacement, vous appuyez sur le bouton FEU. Les touches du clavier restent opérantes comme dans la première version.

```
1 REM      TELECRAN
2 REM  avec  JOYSTICK
3 REM
10 MODE 1
20 X=0:Y=399:PLOT X,Y,1
30 A=ASC(INKEY$+CHR$(0)):C=JOY(0):B=0:M=0:IF A=0
 AND C=0 GOTO 30
32 IF C>=16 THEN B=1
35 IF A>=244 THEN B=1:A=A-4
37 IX=0 : IY=0
40 IF A=243 OR (C AND 8) THEN IX=1 :M=1
50 IF A=241 OR (C AND 2) THEN IY=-1:M=1
60 IF A=242 OR (C AND 4) THEN IX=-1:M=1
70 IF A=240 OR (C AND 1) THEN IY=1 :M=1
75 IF M=1 GOTO 110
80 IF A=16 GOTO 10
85 IF A=127 THEN PLOT X,Y,0 :GOTO 20
90 IF A=224 THEN PLOT X,Y,0 :X=320:Y=200:GOTO 15
0
95 IF A=32  THEN PLOT X,Y,0 :X=639:Y=0  :GOTO 15
0
100 GOTO 30
110 LX=(IX+1)*319.5:LY=(IY+1)*199.5:IF (X=LX) OR
 (Y=LY) GOTO 30
120 IF B=1 GOTO 140
130 PLOT X,Y,0
140 X=X+IX:Y=Y+IY
150 PLOT X,Y,1:GOTO 30
```

UTILISATION DE LA MÉMOIRE SUPPLÉMENTAIRE DU 6128

Nous traitons cette question dans ce livre car cette mémoire est gérée exactement comme un périphérique, notamment comme un disque virtuel.

La première chose à savoir est que vous pouvez utiliser vos 64 K supplémentaires grâce à un logiciel. Donc, avant toute chose, il faut charger et activer ce logiciel, ce qui se fait par la commande :

RUN "BANKMAN"

Le logiciel s'intitule BANKMAN, abréviation de Bank Manager (= Gestionnaire de Bancs Mémoire : la mémoire de 64 K supplémentaires est en effet considérée comme formée de 4 bancs de 16 K). Il réside sur la face 1 de votre première disquette système.

Le logiciel offre deux types de fonctions :

1. – gestion d'écrans multiples, permettant de permuter rapidement deux images ;

2. – simulation d'une disquette en mémoire avec accès direct.

Comme toute instruction rajoutée à BASIC, les commandes commencent par ¦ (SHIFT @).

Ecrans multiples

On dispose de 5 blocs de mémoire d'écran : le bloc 1 est en mémoire normale : c'est celui qui est en cours d'affichage. Les blocs 2 à 5 sont en mémoire supplémentaire : ils servent uniquement de sauvegarde ; pour afficher effectivement l'écran 3, par exemple, il faut le copier sur le bloc 1.

La principale commande est :

¦SCREENCOPY,n1,n2

où n1 et n2 sont des numéros d'écran (de 1 à 5) : on copie l'écran n2 sur le bloc n1. Par exemple, pour sauvegarder l'écran visible sur le bloc 3, vous faites ¦SCREENCOPY,3,1. Pour rendre visible l'écran 4, vous feriez ¦SCREENCOPY,1,4.

Comment faire pour avoir un dessin animé formé de 4 scènes qui se succèdent rapidement de façon cyclique ?

– d'abord il faut préparer les quatre scènes, une par une sur l'écran. Vous pouvez utiliser le logiciel de dessin avec stylo lumineux, ou le TELECRAN ci-dessus.

– lorsqu'une scène est prête, vous la sauvez sur le bloc N (de 2 à 5) par ¦SCREENCOPY,N,1.

– il suffit maintenant d'exécuter le programme :

```
10 FOR I=2 TO 5
20 ¦SCREENCOPY,1,I
30 GOSUB 100
40 NEXT
50 GOTO 10
```

En 100, vous avez un sous-programme qui marque un petit délai, à régler à votre convenance.

Il y a un moyen de faire pareil, mais avec 5 scènes. En effet, il existe une autre instruction :

¦SCREENSWAP,n1,n2

qui, elle, effectue l'*échange* des écrans n1 et n2 (de 1 à 5).

Il suffit alors de remplacer ¦SCREENCOPY par ¦SCREENSWAP,1,I dans notre programme de dessin animé.

Disque virtuel

On dispose d'instructions qui simulent la lecture et l'écriture d'enregistrements en accès direct sur disque : les opérations se font en réalité dans les 64 K supplémentaires. On va voir que les opérations sont très semblables à celles que nous avons fournies dans notre système d'accès direct au chapitre 3.

Bien sûr, il faut démarrer par un RUN "BANKMAN".

Ensuite, il faut "ouvrir" le "fichier" par :

¦BANKOPEN,n

où n est le nombre de caractères par "enregistrement".

Les lectures et les écritures se font respectivement par les instructions :

¦BANKREAD,@R%,X$,NE et ¦BANKWRITE,@R%,X$,NE

où R% est une variable entière qui contiendra 0 si l'opération s'est bien passée et sera négative sinon, X$ est la chaîne de caractères qui constitue l'enregistrement et NE est le numéro d'enregistrement voulu de 0 à concurrence des 64 K. Si NE n'est pas utilisé, on traite le "fichier" en séquentiel.

Il y a aussi une instruction de recherche :

¦BANKFIND,@R%,T$,N1,N2

qui recherche la chaîne T$ dans le fichier entre les enregistrements N1 et N2 (si N1 et N2 ne sont pas spécifiés, la recherche a lieu dans tout le fichier) ; R% est le code de retour : à l'issue de l'instruction, il a pour valeur le numéro de l'enregistrement où se trouve la chaîne cherchée ; si R% vaut −3, la chaîne cherchée n'a pas été trouvée.

Exemple

Le programme qui suit fabrique des permutations de la phrase "Belle Marquise vos beaux yeux me font mourir d'amour".

```
10 REM Faire RUN "BANKMAN" avant de charger
20 |BANKOPEN,8 : R%=0
30 X$=" BELLE   ":|BANKWRITE,@R%,X$,1
31 X$="MARQUISE":|BANKWRITE,@R%,X$,2
32 X$="  VOS    ":|BANKWRITE,@R%,X$,3
33 X$=" BEAUX   ":|BANKWRITE,@R%,X$,4
34 X$=" YEUX    ":|BANKWRITE,@R%,X$,5
35 X$="  ME     ":|BANKWRITE,@R%,X$,6
36 X$=" FONT    ":|BANKWRITE,@R%,X$,7
37 X$=" MOURIR  ":|BANKWRITE,@R%,X$,8
38 X$="D'AMOUR  ":|BANKWRITE,@R%,X$,9
40 INPUT "Permutation ";A$
50 FOR I=1 TO 9
60 K=VAL(MID$(A$,I,1))
70 |BANKREAD,@R%,X$,K : PRINT X$;
80 NEXT : PRINT
90 GOTO 40
```

Si, lorsque le programme vous demande une permutation, vous répondez, par exemple 345129678, vous obtenez VOS BEAUX YEUX BELLE MARQUISE D'AMOUR ME FONT MOURIR.

Nous voici maintenant arrivés au terme de ce livre. Nous avons découvert un grand nombre de périphériques connectables à l'Amstrad qui en améliorent considérablement les performances et lui ouvrent des domaines d'applications très vastes.

Tous les programmes que nous avons cités dans le livre concourent à montrer la simplicité du maniement des périphériques.

Bien sûr, ils sont améliorables et adaptables à chaque application. Nous espérons que le lecteur est maintenant en mesure de le faire.

Annexe 1
MESSAGES D'ERREUR CONCERNANT LES FICHIERS OU PÉRIPHÉRIQUES

Erreurs sous BASIC

Rappelons qu'on peut garder le contrôle après une erreur dans un programme BASIC si l'on a pris la précaution d'inclure un ON ERROR GOTO n. A partir de la ligne n, vous écrivez votre propre routine de traitement d'erreur, terminée par RESUME.

Dans cette routine, vous disposez des paramètres ERR et ERL pour connaître respectivement le numéro de l'erreur et la ligne où elle s'est produite.

Voici les numéros d'erreurs concernant les fichiers, ainsi que la nature de l'erreur en cause.

24 EOF met

On a rencontré la fin de fichier.

25 File type error

Type de fichier erroné ; avec OPENIN, on ne peut lire qu'un fichier ASCII ; avec LOAD, on ne peut lire qu'un fichier produit par SAVE.

27 File already open

Fichier déjà ouvert. Avant OPENIN ou OPENOUT, il faut fermer le fichier ouvert précédemment.

31 File not open

Fichier non ouvert.

32 Broken in ...

Arrêt en ...

Erreurs disquettes

Lorsque ERR vaut 31 ou 32, la variable système DERR donne un complément d'informations. Voici les valeurs possibles avec leur signification :

DERR	Signification
0 ou 22	On a appuyé sur ESC.
14 ou 142	Etat du canal non valable.
15 ou 143	Fin de fichier physique.
16 ou 144	Commande ou nom de fichier incorrect.
17 ou 145	Le fichier existe déjà.
18 ou 146	Le fichier n'existe pas.
19 ou 147	Catalogue plein (maximum 64 places).
20 ou 148	Disquette (en fait, face) pleine.
21 ou 149	La disquette a été changée alors qu'un fichier était ouvert.
22 ou 150	Fichier en lecture seule.
26 ou 154	Fin de fichier logicielle.
----	-----------
64 ou 192 +	Erreurs du contrôleur.
----	-----------
1	Adresse manquante.
2	Disquette protégée en écriture.
4	Secteur introuvable (ou non formaté).
8	Unité non prête ou pas de disquette.
16	Surcharge.
32	Erreur de donnée.

Messages AMSDOS en direct

La plupart des messages suivants (comme disquette absente) sont suivis de la question : **Retry, Ignore or Cancel** ? (Réessayer, Ignorer ou C pour abandonner.) Une fois l'erreur corrigée (par exemple, une fois que vous avez mis une disquette), vous tapez R pour réessayer. Si vous ne pouvez pas corriger, tapez C. Il n'est pas recommandé de taper I. (Vous pouvez taper les lettres en minuscules.)

Voici les messages :

Unknown command

Commande inconnue : commande mal orthographiée, ou, sur 464, oubli de brancher l'interface disquette.

Bad command

Commande mal orthographiée.

"nom" already exists

Le fichier "nom" existe déjà (se produit uniquement pour un nouveau nom dans ¦REN : dans les autres cas, le système crée un fichier .BAK).

"nom" not found

Le fichier "nom" n'est pas trouvé. Ce message est grave pour OPENIN, mais sans importance pour ¦ERA, par exemple.

Drive x: directory full

Le catalogue est plein sur la disquette x:. Il n'y a place que pour 64 fichiers.

Drive x: disc full

La disquette (ou plutôt la face) x: est pleine.

Drive x: disc changed, closing nom

On change de disquette alors que le fichier 'nom' est ouvert. Si c'est en écriture, il y aura des informations perdues.

Nom is read only

Le fichier 'Nom' est en lecture seule. Cet attribut ne peut être donné que sous CP/M (par STAT ou SET) mais AMSDOS le respecte.

Drive x: disc missing

Pas de disquette dans l'unité x: ou elle a été mal enfoncée, ou elle n'est pas formatée. Remédier au problème en mettant une disquette formatée (et en la mettant bien) et faire R. Si le problème ne se résout pas, il y a peut-être une panne.

Drive x: disc is write protected

Essayez d'écrire sur une disquette protégée en écriture. Si c'est une disquette sur laquelle il est loisible d'écrire, sortez-la de l'unité et positionnez convenablement le taquet de protection ; remettez-la dans l'unité et faites R.

Drive x: read fail

On n'arrive pas à lire sur la disquette x. Essayez de la sortir et de la remettre. Si le problème ne se résout pas, c'est peut-être une panne.

Drive x: write fail

On n'arrive pas à écrire sur la disquette x. Faites comme ci-dessus.

Fail to load CP/M

Echec de la commande ¦CPM. Vérifiez que c'est bien une disquette comportant le système CP/M qui est montée.

Annexe 2
QUESTIONS ET RÉPONSES

Je ne comprends pas pourquoi, sur cassette ou disquette, il faut écrire une variable à la fois.

Ce n'est pas un impératif absolu. C'est le moyen d'escamoter un problème très grave. Supposons que vous ayez fait :

```
PRINT#9,A$;B$
```

Lorsque vous voudrez relire par :

```
INPUT#9,A$,B$
```

le système va recevoir une chaîne de caractères unique. Pour pouvoir séparer A$ et B$, il faut (règles de INPUT) que BASIC trouve une virgule ou un caractère retour-chariot (CHR$(13)) entre les deux chaînes.

Il faudrait donc écrire :

```
PRINT#9,A$;",";B$
```

avec la virgule fournie explicitement.

Lorsqu'on écrit les variables une par une :

```
PRINT#9,A$ : PRINT#9,B$
```

le système incorpore automatiquement un retour-chariot entre chaque variable. A la lecture, on lit une variable à la fois et le retour-chariot délimite chaque variable.

Notons que l'emploi de WRITE#9,A$,B$ offrirait une autre solution à ce problème.

Pourquoi, sur disquette, faut-il explicitement gérer le retour-chariot en fin d'écriture ? Quelle est la différence entre PRINT#9,A$ et PRINT#9,A$;CHR$(13); ?

Dans PRINT#9,A$;CHR$(13); il y a en tout et pour tout un retour-chariot écrit par le système. Dans PRINT#9,A$ il y a un retour-chariot (CHR$(13)) plus une alimentation-ligne (CHR$(10)) écrits par le système. Le caractère alimentation-ligne trouble la relecture sur disquette.

J'ai entendu parler de termes tels que "sécurité" ou "intégrité" dans les bases de données. De quoi s'agit-il ?

Il s'agit de problèmes importants et délicats. L'intégrité de la base de données fait référence à la nécessité que la base de données soit *cohérente* à tout instant. Par exemple, si l'adresse d'un client se trouve dans deux des fichiers de la base, il faut s'assurer qu'en cas de changement d'adresse de ce client, la nouvelle adresse soit écrite à ces deux endroits, y compris dans le cas où une panne de courant interviendrait entre les deux écritures.

Un cas, qui n'est pas à envisager pour le moment sur les micro-ordinateurs familiaux, est celui de l'accès simultané à la base à partir de plusieurs terminaux : il ne faudrait pas alors qu'un utilisateur puisse changer un enregistrement pendant qu'un autre cherche à le lire.

La sécurité des données recouvre trois choses. Il faut d'abord protéger les données contre des accès dangereux :

– en lecture : il faut empêcher que des données soient consultées par des indiscrets ;

– en écriture : il faut empêcher que des données soient altérées par qui n'est pas qualifié pour le faire.

Pour éviter cela, il faut établir des systèmes de mots de passe, de clés d'accès personnalisées, etc. CP/M Plus permet d'employer des mots de passe.

Le deuxième problème concerne la protection des données contre la destruction : protection-écriture des disquettes, copies de sécurité, rangement en armoires ignifugées... Voir aussi les précautions de manipulation indiquées aux chapitres 3 et 4.

Le troisième point, très important aussi, est qu'il faut protéger les données contre l'erreur, c'est-à-dire qu'il ne faut pas que la base contienne des données fausses.

C'est surtout au niveau de l'introduction (la saisie) des données que les erreurs se produisent le plus souvent. Les programmes de saisie doivent être très soignés pour essayer de déceler, à la source, toute faute de frappe ou toute incohérence des données.

Dans un programme, je veux que les impressions aillent au choix sur l'imprimante ou sur l'écran. Comment faire pour que certaines impressions

aillent dans tous les cas à l'écran, tandis que d'autres vont, au choix à l'écran ou à l'imprimante ?

Utilisez PRINT pour les impressions qui vont dans tous les cas vers l'écran. Pour les autres, utilisez PRINT#IM,... la variable IM ayant reçu la valeur 0 ou 8 au moment du choix.

Je veux rendre une rubrique muette dans un enregistrement de la base de données, par exemple en ne tapant que des blancs. Mais cela ne fonctionne pas.

Au lieu de mettre des blancs, vous pouvez mettre un ? ce qui correspond assez bien à la réalité : si vous voulez rendre une rubrique muette, c'est parce que vous ignorez la donnée correspondante pour l'enregistrement concerné. Vous pourrez ultérieurement compléter par une mise à jour lorsque vous connaîtrez la donnée.

Dans le système de gestion de fichiers du chapitre 3, est-on limité à une seule disquette ?

Avec quelques aménagements au programme, un fichier global pourrait s'étendre sur plusieurs disquettes, ce qui élargirait beaucoup la taille des applications traitables, mais il faudrait compter avec la lenteur des échanges de disquettes.

La gestion du fichier résumé serait légèrement modifiée : il en faudrait un exemplaire sur chacune des disquettes, comportant l'information sur l'intervalle d'enregistrements présents sur cette disquette. Dans la routine de lecture d'un bloc, il faudrait un message demandant le montage de la disquette convenable, si ce n'est pas elle qui est en place.

On pourrait aussi utiliser deux unités de disquettes, un enregistrement étant cherché sur la B s'il n'est pas trouvé sur la A.

Annexe 3
SOLUTION DES EXERCICES

Exercice 1.1.

```
10 INPUT "Cassette (C) ou Disquette (D)";P$
20 IF P$="C" OR P$="c" THEN ¦TAPE ELSE ¦DISC
30 OPENIN "nom"
```

Exercice 1.2.

```
10 REM RELECTURE D'UN FICHIER
20 OPENIN "CARRES"
30 FOR M I=1 TO 50
40 INPUT#9,K
50 PRINT I,K : NEXT
60 CLOSEIN
```

Exercice 1.3.

```
100 OPENOUT "CHAINES"
110 FOR I=1 TO 10
120 WRITE#9,A$(I)
130 NEXT
140 CLOSEOUT
```

Exercice 1.4.

```
10 REM RELECTURE D'UN FICHIER
20 OPENIN "CARRES"
30 WHILE NOT EOF
```

```
40 INPUT#9,K
50 PRINT I,K
60 WEND
70 CLOSEIN
```

Exercice 1.5.

On ajoute :

```
107 INPUT "CODE3:X, Y OU Z";A$
108 PRINT#9,A$;CR$;
```

Exercice 1.6.

Après chaque instruction de numéro n=30, 40, 50 et 120, ajouter :

```
[n+1] A$=LEFT$(A$,20)
```

Si LEN(A$)<20, A$ est conservé tel quel, sinon seuls les 20 premiers caractères sont conservés.

Si l'on veut que les rubriques aient toujours exactement 20 caractères, on fait :

```
A$=LEFT$(A$+STRING$(20," "),20).
```

Exercice 1.7.

30 devient :

```
30 INPUT "NUMEROS,VALEURS";K,K$,L,L$
```

et, entre 75 et 90, on a les lignes :

```
80 IF K=0 GOTO 85
82 IF A$(K)<>K$ GOTO 50
85 IF L=0 GOTO 90
87 IF A$(L)<>L$ GOTO 50
```

On a ici réalisé la **et** les deux conditions. Exercez-vous à réaliser le **ou** (lister l'enregistrement *dès que l'une* des conditions est satisfaite).

Exercice 1.8.

30 devient :

```
30 INPUT "NUMERO,DEBUT,NB.CAR,VALEUR";K,DE,N,K$
```

et 85 devient :

```
85 IF MID$(A$(K),DE,N)<>K$ GOTO 50
```

Exercice 2.1.

```
10 INPUT "ECRAN (E) OU IMPRIMANTE (I)";IM$
20 IF IM$="I" THEN IM=8 ELSE IM=0
```

et l'on écrit les sorties PRINT#IM, ...

Exercice 2.2.

```
10 PRINT#8,"NOMBRE","CARRE","NOMBRE","CARRE"
20 FOR I=1 TO 50
30 PRINT#8,I,I*I,I+50,(I+50)*(I+50)
40 NEXT
```

Exercice 2.3.

30 devient :

```
30 PRINT#8,USING "#####        ";I,I*I,I+50,(I+50)*(I+50)
```

(Attention aux ',' et ';'. Il y a 5 # et 8 espaces.)

Exercice 2.4.

Voici l'impression produite et le programme correspondant :

TABLE DE SIN X

X	SIN X	X	SIN X	X	SIN X	X	SIN X	X	SIN X
.00	.00000	.01	.01000	.02	.02000	.03	.03000	.04	.03999
.05	.04998	.06	.05996	.07	.06994	.08	.07991	.09	.08988
.10	.09983	.11	.10978	.12	.11971	.13	.12963	.14	.13954
.15	.14944	.16	.15932	.17	.16918	.18	.17903	.19	.18886
.20	.19867	.21	.20846	.22	.21823	.23	.22798	.24	.23770
.25	.24740	.26	.25708	.27	.26673	.28	.27636	.29	.28595
.30	.29552	.31	.30506	.32	.31457	.33	.32404	.34	.33349
.35	.34290	.36	.35227	.37	.36162	.38	.37092	.39	.38019
.40	.38942	.41	.39861	.42	.40776	.43	.41687	.44	.42594
.45	.43497	.46	.44395	.47	.45289	.48	.46178	.49	.47063
.50	.47943	.51	.48818	.52	.49688	.53	.50553	.54	.51414
.55	.52269	.56	.53119	.57	.53963	.58	.54802	.59	.55636
.60	.56464	.61	.57287	.62	.58104	.63	.58914	.64	.59720
.65	.60519	.66	.61312	.67	.62099	.68	.62879	.69	.63654
.70	.64422	.71	.65183	.72	.65938	.73	.66687	.74	.67429
.75	.68164	.76	.68892	.77	.69614	.78	.70328	.79	.71035

```
1 REM
2 REM  Exercice 2-4
3 REM
10 PRINT#8,SPC(25);"TABLE DE SIN X"
20 PRINT#8:PRINT#8
30 PRINT#8,STRING$(76,"-")
40 B$="!   X   ! SIN X   "
50 FOR I=1 TO 5 :PRINT#8,B$; : NEXT
60 PRINT#8,"!"
70 PRINT#8,STRING$(76,"-")
80 FOR X1=0 TO PI/4 STEP 0.05
90 FOR X=X1 TO X1+0.041 STEP 0.01
100 Y=SIN(X)
110 PRINT#8,"!";USING " .## ";X;
120 PRINT#8,"!";USING " .##### ";Y;
130 NEXT
140 PRINT#8,"!"
150 NEXT
160 PRINT#8,STRING$(76,"-")
```

Exercice 2.5.

```
ALPHA = α
```

```
1 REM Exercice 2-5
10 DATA 27,75,0,6,28,34,34,28,34,0
20 AL$=""
30 FOR I=1 TO 10 : READ AL
40 AL$=AL$+CHR$(AL) : NEXT
50 PRINT#8,"ALPHA = ";AL$+CHR$(15)
```

Exercice 2.6.

```
10 REM Exercice 2.6
20 G$=CHR$(27)+CHR$(75)+CHR$(0)+CHR$(120)
30 A$=STRING$(20,103) : B$=STRING$(20,123)
40 WIDTH 255
50 PRINT#8,G$;A$;B$;A$;B$;A$;B$;CHR$(15)
```

Exercice 2.7.

```
120
90
40
75
5
8
53
68
110
30
```

```
1 REM
2 REM  Exercice 2-7
3 REM
10 A$=CHR$(127):R$=CHR$(28):CN$=CHR$(15)
20 WIDTH 255
30 FOR I=1 TO 10
40 READ N : IF N<128 GOTO 70
50 N=N-127 : N$=CHR$(127)
60 PRINT#8,R$;N$;A$;
70 N$=CHR$(N)
80 PRINT#8,R$;N$;A$;CN$;"    ";N
90 NEXT
1000 DATA 120,217,40,75,5,135,180,195,110,30
```

Exercice 2.8.

```
120
217
 40
 75
  5
135
180
195
110
 30
```

```
1 REM
2 REM  Exercice 2-8
3 REM
10 A$=CHR$(127):R$=CHR$(28):CN$=CHR$(15)
```

```
20 WIDTH 255
30 FOR I=1 TO 10
40 READ N : PRINT#8,USING "###   ";N; : IF N<128
 GOTO 70
50 N=N-127 : N$=CHR$(127)
60 PRINT#8,R$;N$;A$;
70 N$=CHR$(N)
80 PRINT#8,R$;N$;A$;CN$
90 NEXT
1000 DATA 120,217,40,75,5,135,180,195,110,30
```

Exercice 2.9.

```
J'insiste sur ce point
J' i n s i s t e sur ce point
J'insiste sur ce point
  --------
```

```
1 REM
2 REM Exercice 2-9
3 REM
10 PRINT#8,"J'insiste sur ce point";CHR$(20);
20 PRINT#8,"  insiste ";CHR$(20);
30 PRINT#8,"  insiste " ' Superposition
40 PRINT#8,"J'";CHR$(14);"insiste";CHR$(15);" su
r ce point" ' Elargissement
50 PRINT#8,"J'insiste sur ce point"
60 PRINT#8,"  --------" ' Soulignement
```

Exercice 2.10.

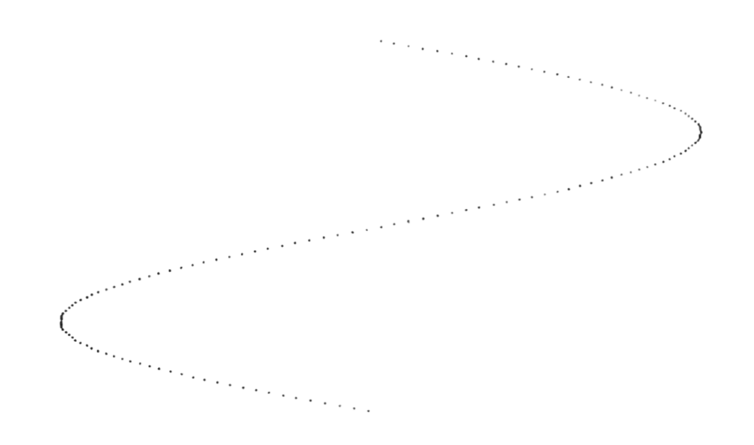
```
1 REM
2 REM Exercice 2-10  2e version
3 REM Copie Ecran H-R DMP 1
4 REM
10 WIDTH 255 : G$=CHR$(27)+CHR$(75)+CHR$(1)+CHR$
(72)
20 FOR X=0 TO 639 STEP 14
30 A$="" : FOR Y=0 TO 399 STEP 2
40 N=0 : FOR I=0 TO 6
50 N=N-(2^I)*(TEST(X+I+I,Y)<>0)
```

```
60 NEXT : A$=A$+CHR$(N)
70 NEXT
80 PRINT#8,G$;A$
90 NEXT
100 PRINT#8,CHR$(15)
```

Le programme est indépendant du mode d'affichage. Il faut une quarantaine de minutes pour avoir un écran. La seule solution est l'emploi du langage machine. Toutefois la version ci-dessous est plus rapide, mais elle risque d'"oublier" des pixels en mode 2.

```
1 REM
2 REM Exercice 2-10
3 REM CoPie Ecran H-R DMP 1
4 REM
10 WIDTH 255 : G$=CHR$(27)+CHR$(75)+CHR$(1)+CHR$
(72)
20 FOR X=0 TO 639 STEP 7
30 A$="" :B$="" : FOR Y=0 TO 199
40 N=0 : M=0 : Z=Y+200 : FOR I=0 TO 6
50 N=N-(2^I)*(TEST(X+I,Y)<>0)
55 M=M-(2^I)*(TEST(X+I,Z)<>0)
60 NEXT : A$=A$+CHR$(N) : B$=B$+CHR$(M)
70 NEXT
80 PRINT#8,G$;A$;G$;B$
90 NEXT
100 PRINT#8,CHR$(15)
```

Exercice 2.11.



```
1 REM
2 REM Exercice 2-11
3 REM
10 DIM M(479) : G$=CHR$(27)+CHR$(76)+CHR$(96)+CH
R$(0)
20 WIDTH 255 :PRINT#8,CHR$(27)+CHR$(49)
30 DEF FNF(X)=INT(240+200*SIN(PI*X/70))
40 FOR X=0 TO 139 STEP 7
50 FOR K=0 TO 479 : M(K)=0 : NEXT
60 FOR L=0 TO 6 : K=FNF(X+6-L)
70 M(K)=M(K) OR 2^L
80 NEXT
90 FOR K=0 TO 479 STEP 96 : PRINT#8,G$; :FOR L=0
 TO 95 : PRINT#8,CHR$(M(K+L)); : NEXT :NEXT
100 PRINT#8 : NEXT
```

Exercice 2.12.

Dans la méthode des caractères définis par l'utilisateur, une fois que le caractère est défini, il n'y a qu'un octet à transmettre à l'imprimante à chaque fois qu'on veut l'imprimer, alors qu'en mode graphique, il faut transmettre au moins 6 octets à chaque fois. La deuxième méthode est donc avantageuse si l'on veut imprimer le caractère un très grand nombre de fois.

Exercice 2.13.

On ajoute les instructions suivantes :

```
27 INPUT "No Rubrique, Valeur recherchee";K,K$

95 IF K=0 GOTO 100
97 IF A$(K)<>K$ GOTO 50
```

Exercice 3.1.

B:PROG.* : fichier sur la disquette B, de nom PROG, d'extension quelconque, éventuellement sans extension.

*.BAS : fichier de nom quelconque, d'extension .BAS, donc n'importe quel programme BASIC.

P*.B* : Fichier de nom commençant par P et extension commençant par B.

Exercice 3.2.

???ZOZO*.*

Exercice 3.3.

Les instructions modifiées ou ajoutées sont :

```
3100 IF A$=“<” OR A$=“=” OR A$=“>” OR A$=“#” GOTO 3130
3135 IF (IC>1) AND (A$=“#” OR CC$=“#”) GOTO 3190
3240 IF CC$<>“#” THEN VV$=LEFT$(VV$+“                              ”,
      TA(RR))
15015 IF CC$<>“#” GOTO 15020
15016 IF INSTR(1,R$,VV$)<>0 THEN SAT=1
15017 GOTO 15070
```

Exercice 3.4.

Tous les END doivent être remplacés par RUN. Il y en a en 1140, 2190, 3190, 3230, 3330, 4100, 4280, 5130 et 6370 (en prenant en compte les fonctions ajoutées dans la suite du chapitre).

Ensuite, on introduit le début du programme :

```
1 REM
2 REM MENU
3 REM
100 PRINT:PRINT:PRINT:PRINT
110 PRINT"Tapez"
120 PRINT"      C pour Creation"
130 PRINT"      A pour Ajout"
140 PRINT"      L pour Liste selective"
150 PRINT"      T pour Tri"
160 PRINT"      M pour Modification"
170 PRINT"      J pour mise a Jour"
180 A$=INKEY$ : IF A$="" GOTO 180
190 IF A$="C" THEN RUN 1000
200 IF A$="A" THEN RUN 2000
210 IF A$="L" THEN RUN 3000
220 IF A$="T" THEN RUN 4000
230 IF A$="M" THEN RUN 5000
240 IF A$="J" THEN RUN 6000
250 GOTO 110
```

Exercice 3.5.

Oui, à part quelques imperfections pour lesquelles il est facile d'apporter des compléments.

On crée un fichier avec un enregistrement pour chaque jour. Le premier libellé sera "DATE", ensuite "JOUR", puis des rubriques du style "AVANT 8H", "9H-10H" ..."11H-12H", ..."18H-19H", "APRES 19H" et enfin une rubrique "NOTES".

Pour connaître l'emploi du temps de tel jour, il suffit d'une liste avec sélection sur DATE=...

Pour changer un rendez-vous, il faut faire une modification et il n'y aura jamais de rature.

Il serait utile de disposer d'une recherche générale du type trouver une chaîne de caractères quelconque au milieu d'une rubrique non précisée : cela résoudrait le problème "quand ai-je rendez-vous avec M. Dupont?".

Exercice 3.6.

Oui. Les rubriques seront du style DATE, DESTINATAIRE/EXPEDITEUR, SOMME, RENTREE, etc. Il faut utiliser la fonction cumul des sommes des chèques émis non encore débités.

Exercice 3.7.

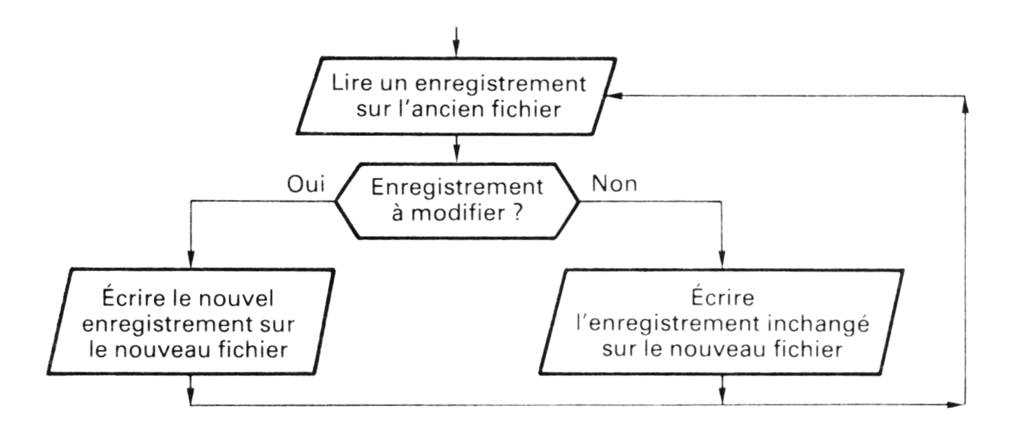
```
...
7050 S=0 : FP=0
7060 FOR I=1 TO NS
7070 E=IP(I)
7080 GOSUB 12000 ' Lecture Bloc.
7090 S=S+VAL(MID$(ENR$(EE),D(R),TA(R)))
7100 NEXT
7110 PRINT "CUMUL = ";S
...
```

Exercice 3.8.

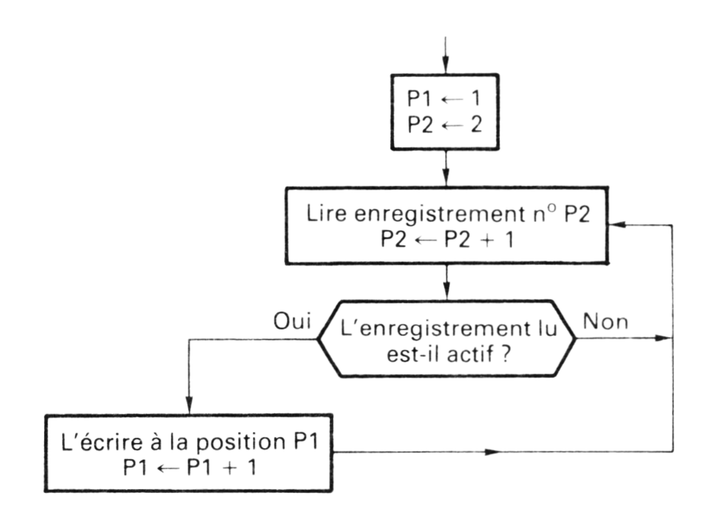
On peut remplacer les instructions 6320 à 6360 par la simple instruction :

```
6320 |ERA,NOM$+"?*.a"
```

Exercice 4.1.

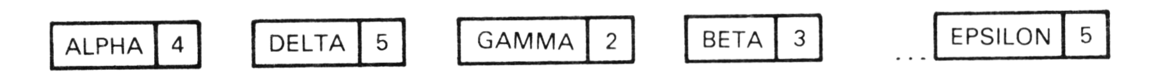


Exercice 4.2.



Nous n'avons pas fait ici figurer de traitement de la fin de fichier, mais il ne faudrait pas l'oublier.

Exercice 4.3.



L'enregistrement EPSILON reste dans le fichier mais il n'est plus possible de l'atteindre puisque le pointeur qui pointait sur lui a été modifié.

Annexe 4
INDEX DES COMMANDES DES PÉRIPHÉRIQUES

Le nombres d'étoiles marque l'importance de la commande (mais n'hésitez pas à utiliser une commande "peu importante" si elle est intéressante pour vous). La page où la commande est étudiée figure dans la colonne de droite.

Cassettes

			Page
***	LOAD "nom"	Charge le programme 'nom' à partir de la cassette.	15
***	LOAD NOM$	Le nom peut être spécifié sous forme d'une variable.	15
*	LOAD "!nom"	Charge le programme 'nom' sans afficher les messages de démarrage du magnétophone.	14
***	SAVE "nom"	Sauve le programme BASIC avec le nom 'nom'.	13
	SAVE "nom",A	Sauve le programme sous forme ASCII.	13

			Page
	SAVE "nom",P	Sauve le programme sous forme protégée.	13
**	SAVE "nom",B,ad,n	Sauve la zone binaire de n octets, adresse de départ ad.	13
**	RUN "nom"	Charge et exécute le programme 'nom'.	15
*	CHAIN "nom"	Charge et exécute le programme 'nom' sans perdre les variables.	18
*	MERGE "nom"	Fusionne le programme 'nom' avec le programme déjà en mémoire.	15
	CHAIN MERGE "nom"	Fusionne et exécute le programme 'nom'.	18
	CHAIN MERGE "nom", l,DELETE m-n	Fusionne et exécute à partir de la ligne l le programme 'nom' après avoir détruit les lignes m à n.	18
	SPEED WRITE n	Fixe la vitesse du magnétophone ; n=1 rapide ; n=0 lent.	14
***	CAT	Fait afficher la liste des fichiers.	14
*** ***	OPENIN "nom" OPENIN NOM$	Ouvre en lecture le fichier 'nom'.	19

			Page
*** ***	OPENOUT "nom" OPENOUT NOM$	Ouvre en écriture le fichier 'nom'.	19
***	INPUT#9,A$	Lit la variable A$ sur cassette.	21
	LINE INPUT#9,A$	Lit toutes les informations jusqu'au retour-chariot.	26
***	PRINT#9,A$;CR$;	Écrit sur cassette la variable A$ suivie d'un retour-chariot.	21
*	WRITE#9,A,B	Écrit sur cassette avec virgules explicites.	26
***	CLOSEIN	Ferme le fichier en lecture.	23
***	CLOSEOUT	Ferme le fichier en écriture.	23
**	¦TAPE	Dirige les entrées-sorties sur cassette.	11
*	¦TAPE.IN	Dirige les entrées sur cassette.	11
*	¦TAPE.OUT	Dirige les sorties sur cassette.	11

Disquettes

			Page
***	LOAD "nom"	Charge le programme 'nom' à partir de la disquette.	15
***	LOAD NOM$	Le nom peut être spécifié sous forme d'une variable.	15
***	SAVE "nom"	Sauve le programme BASIC avec le nom 'nom'.	13
	SAVE "nom",A	Sauve le programme sous forme ASCII.	13
	SAVE "nom",P	Sauve le programme sous forme protégée.	13
**	SAVE "nom",B,ad,n	Sauve la zone binaire de n octets, adresse de départ ad.	13
**	RUN "nom"	Charge et exécute le programme 'nom'.	15
*	CHAIN "nom"	Charge et exécute le programme 'nom' sans perdre les variables.	18
*	MERGE "nom"	Fusionne le programme 'nom' avec le programme déjà en mémoire.	15
	CHAIN MERGE "nom"	Fusionne et exécute le programme 'nom'.	18

			Page
	CHAIN MERGE "nom", I,DELETE m-n	Fusionne et exécute à partir de la ligne I le programme 'nom' après avoir détruit les lignes m à n.	18
***	CAT	Fait afficher la liste des fichiers.	69
*** ***	OPENIN "nom" OPENIN NOM$	Ouvre en lecture le fichier 'nom'.	19
*** ***	OPENOUT "nom" OPENOUT NOM$	Ouvre en écriture le fichier 'nom'.	19
***	INPUT#9,A$	Lit la variable A$ sur disquette.	21
	LINE INPUT#9,A$	Lit toutes les informations jusqu'au retour-chariot.	26
***	PRINT#9,A$;CR$;	Écrit sur disquette la variable A$ suivie d'un retour-chariot.	21
*	WRITE#9,A,B	Écrit sur cassette avec virgules explicites.	26
***	CLOSEIN	Ferme le fichier en lecture.	23
***	CLOSEOUT	Ferme le fichier en écriture.	23
**	¦DISC	Dirige les entrées-sorties sur disquette.	11 69

			Page
*	¦DISC.IN	Dirige les entrées sur disquette.	71
*	¦DISC.OUT	Dirige les sorties sur disquette.	71

COMMANDES AMSDOS

			Page
	¦A	Active la disquette A.	71
	¦B	Active la disquette B.	71
**	¦CPM	Appelle CP/M.	68
***	¦DIR	Affiche le catalogue.	71
***	¦DIR,"nom"	Affiche le catalogue des fichiers compatibles avec 'nom'.	71
	¦DRIVE,"d"	Active la disquette d.	71
**	¦ERA,"nom"	Efface les fichiers compatibles avec 'nom'.	72
*	¦REN,"nouveau", "ancien"	Change le nom du fichier 'ancien' en 'nouveau'.	72

			Page
	¦USER,n	Active l'utilisateur n.	72
	(Commandes ¦TAPE et ¦DISC pour mémoire.)		

COMMANDES CP/M

(+ = CP/M Plus ; 2 = CP/M 2.2)

			Page
***	AMSDOS	Repasse à AMSDOS/BASIC.	68
	ASM	Appelle l'Assembleur 8080.	96
2	BOOTGEN	Copie CP/M et variables système.	93
2	CHKDISC	Comparaison de disquettes (système à deux unités).	68
2*	CLOAD "C" D	Copie le fichier cassette C sur disquette sous le nom D.	94
2**	COPYDISC	Copie de disquette entière (système à deux unités).	68
2*	CSAVE D "C"	Copie le fichier disquette D sur cassette sous le nom C.	94

			Page
	DDT	Aide à la mise au point.	96
+	DEVICE	Affiche ou impose le mode de fonctionnement de périphériques.	96
***	DIR	Catalogue de la disquette.	70
***	DIR nf	Catalogue des fichiers compatibles avec nf.	92
+	DIR [FULL]	Catalogue très complet.	92
+	DIRS ou DIRSYS	Liste des fichiers système.	93
2	DISCCHK	Comparaison de disquettes (système à une unité).	68
2	DISCCOPY	Copie de disquette entière (système à une unité).	68
+*** 2**	DISCKIT3 DISCKIT2	Menu pour copie, formatage, vérification, etc.	68
	DUMP	Sortie hexadécimale d'un fichier.	97

			Page
*	ED	Éditeur de programmes ou textes.	97
**	ERA nf	Efface le(s) fichier(s) compatible(s) avec nf.	91
2**	FILECOPY nf	Copie de fichier(s) d'un disque à un autre sur la même unité.	91
2***	FORMAT	Formatage de disquette.	68
+	GENGRAF np	Munit le programme np d'extensions graphiques.	96
+	GET nf	Prend l'entrée console sur le fichier nf.	94
+*	HELP	Utilitaire d'aide.	96
+	LANGUAGE n	Active le jeu de caractères nationaux n (France = 1).	95
2	MOVCPM	Déplace CP/M.	93
+	PALETTE f t	Fixe les couleurs de fond et texte.	95
***	PIP dest=source	Copie source sur destination (fichiers ou périphériques).	93

			Page
+	PUT nf	Effectue les sorties système sur le fichier nf.	94
*	REN nouveau=ancien	Change le nom du fichier ancien en nouveau.	91
+	SET nf [opt]	Impose des options (lecture seule mot de passe etc.) au fichier nf.	92
+	SET24X80 ON ou OFF	Met en 24 lignes sur 80.	95
+	SETKEYS nf	Change le rôle des touches du clavier d'après le fichier nf.	95
+	SETLST nf	Envoie les codes de contrôle du fichier vers l'imprimante.	94
+**	SETSIO par	Fixe les paramètres du RS 232 (voir ci-dessous).	95
2	SETUP	Menu de fixation de mode de fonctionnement des périphériques.	
+	SHOW	Affiche l'état du disque.	91
2	STAT	Affiche l'état du disque.	91
2	STAT nf	Affiche les options d'un fichier.	91
2	STAT nf,attr	Fixe les attributs d'un fichier.	91

			Page
**	SUBMIT nf	Exécute la suite de commandes contenues sur le fichier nf.	96
2	SYSGEN *	Génère un système CP/M.	93
**	TYPE nf	Affiche le fichier nf.	94
	USER ou USE n	Active l'utilisateur n.	93

Imprimante

			Page
***	PRINT#8,A,B$;C	Imprime sur l'imprimante.	39
***	PRINT#IM,CHR$(N)	Imprime sur imprimante si IM=8.	39
**	PRINT#8, USING "fmt";A,B,C	Imprime avec format.	40
***	LIST l-m,#8	Listing sur imprimante.	42
**	WIDTH n	Fixe la largeur de ligne.	42

SÉQUENCES DE CONTROLE DE LA DMP 1

			Page
	10	Va à la ligne.	43
***	13	Retour-chariot à la ligne.	43
*	20	Retour-chariot sur la même ligne.	43
**	14	Caractères double largeur.	44
**	15	Retour aux caractères normaux.	44
*	16 n	Positionnement en caractères.	44
	27 16 n n	Positionnement en colonnes élémentaires.	45
***	27 75 n n d...	Mode graphique.	46
*	28 n n d	Répétition d'un motif graphique.	47

SÉQUENCES DE CONTROLE DE LA DMP 2000

			Page
	7	Sonnerie.	58
*	8	Retour arrière.	58
	9	Tabulation horizontale.	58
	10	Va à la ligne.	58
	11	Tabulation verticale.	58
*	12	Va à la page.	58
***	13	Retour-chariot.	58
**	14	Passage en double largeur.	52
**	15	Passage en condensé.	53
**	18	Sortie du mode condensé.	53
**	20	Sortie du mode double largeur.	52
	24	Vide le tampon.	
	127	Supprime le dernier caractère.	
**	27 14	Passage en double largeur.	52
**	27 15	Passage en condensé.	52
	27 33 n	Sélection mode d'impression.	
	27 35	Accepte 8e bit (impossible sur Amstrad).	
	27 37 0 0	Sélectionne les caractères internes.	
*	27 37 1 0	Sélectionne les caractères définis utilisateur.	59
*	27 38 0 d..d	Définit des caractères utilisateur.	59

			Page
	27 42 d...	Mode graphique image bits.	
**	27 45 1/0	Active ou désactive mode souligné.	54
	27 47 n	Sélectionne un canal de tabulation.	
***	27 48	Interligne 1/8 pouce.	56
***	27 49	Interligne 7/72 pouce.	56
**	27 50	Interligne 1/6 pouce.	56
*	27 51 n	Interligne n/216 pouce.	56
***	27 52	Passe en italique.	53
***	27 53	Sort d'italique.	53
*	27 54	Sélectionne jeu de caractères 2.	
*	27 55	Sélectionne jeu de caractères 1.	
	27 56	Inhibe détection de fin de papier.	
	27 57	Active détection de fin de papier.	
	27 58 0 n	Copie le jeu de caractères internes sur le jeu défini utilisateur.	
	27 60	Ramène la tête d'impression à gauche.	
	27 61	Force à 0 le 8e bit.	
	27 62	Force à 1 le 8e bit.	
	27 63 d...d	Change le mode graphique image bits.	
	27 64	Remise à zéro de l'imprimante.	
**	27 65 n	Interligne à n/72 pouce.	56
	27 66 n..n 0	Installe des tabulations verticales.	
*	27 67 n	Fixe la hauteur de page à n lignes.	
	27 67 0 n	Fixe la hauteur de page à n pouces.	
	27 68 n..n 0	Installe des tabulations horizontales.	
***	27 69	Caractères gras.	53
***	27 70	Plus de caractères gras.	53
**	27 71	Mode double frappe.	53
**	27 72	Plus de double frappe.	53
	27 73 1/0	Rend les caractères de contrôle imprimables.	59
	27 74 n	Fixe l'échappement à n/216 pouce.	
***	27 75 n n d...	Mode graphique simple densité.	56
***	27 76 n n d...	Mode graphique double densité.	56
**	27 77	Caractères mini (élite).	53

			Page
	27 78 n	Sauter n lignes à la pliure.	
	27 79	Plus de saut à la pliure.	
**	27 80	Plus de caractères mini.	53
	27 81 n	Fixe la marge droite.	
**	27 82 n	Choisit un jeu de caractères nationaux.	58
***	27 83 1/0	Exposant / indice.	53
***	27 84	Supprime exposant ou indice.	53
	27 85 1/0	Impression unidirectionnelle ou non.	
**	27 87 1/0	Active ou supprime double largeur.	54
**	27 89 n n d...	Mode graphique double densité, vitesse double.	
**	27 90 n n d...	Mode graphique quadruple densité.	
	27 94 n ...	Mode graphique image 9 bits (impossible avec l'Amstrad).	
	27 98 n.. 0	Installe des tabulations dans un canal.	
	27 105 1/0	Active/désactive impression immédiate.	
	27 106 n	Fixe à n/216 pouce l'échappement arrière.	
	27 108 n	Fixe la marge gauche.	
**	27 112 1/0	Active/désactive le mode proportionnel.	53
	27 115 1/0	Active/désactive l'impression lente.	
***	27 120 1/0	Active/désactive la qualité courrier.	53

Interface RS 232

			Page
***	¦SETSIO, ve,vr,c,bd,p,ba	Fixe les paramètres du RS 232 : seul le 1er (vitesse d'émission) est obligatoire.	112

			Page
	¦CLOSESIO	Arrête la communication.	
	¦SETTIMEOUT,n	Fixe le temps d'attente de réponse.	
*	¦SIO,@S%	Obtient l'état de l'interface dans la variable S%.	
	¦RINGWAIT,@S%,n	Attend un appel pendant n sec.	
	¦BREAKSEND,@S%,n	Envoie un signal "Break".	
**	¦INCHAR,@S%,@C%	Reçoit un caractère.	114
**	¦OUTCHAR,@S%,@C%	Envoie un caractère.	114
	¦SETBLOCKEND,n	Fixe le caractère de fin de bloc.	
*	¦INBLOCK,@S%,@X$	Reçoit un bloc.	114
*	¦OUTBLOCK, @S%,@X$	Envoie un bloc.	114
	¦SETFILEEND,n	Spécifie le caractère fin de fichier (défaut 26).	
***	¦INFILE,"nom"	Reçoit le fichier disque 'nom'.	113
***	¦OUTFILE,"nom"	Envoie le fichier disque 'nom'.	113

			Page
*	¦SUCK,"nom"	Reçoit le fichier 'nom' avec correction des erreurs de transmission.	113
*	¦BLOW,"nom"	Envoie le fichier 'nom' avec correction des erreurs de transmission.	113
***	¦SERIAL	Dirige la sortie imprimante (PRINT#8 = imprimante.	112
**	¦PARALLEL	Rétablit #8 = imprimante.	112
	¦XON / ¦NOXON	Restaure/annule la procédure de synchronisation XON/XOFF.	
**	¦TERMINAL	Déguise l'Amstrad en terminal.	114
	¦HALFDUPLEX	Assure l'écho clavier vers écran.	
	¦FULLDUPLEX	Inhibe l'écho clavier vers écran.	
	¦CTRLDISPLAY	Affiche les caractères de contrôle.	
	¦CTRLACTION	Obéit aux caractères de contrôle.	
	¦PRESTEL,@C%	Envoie le caractère à un serveur PRESTEL.	
	¦SAVEPRESTEL,"nom"	Sauve sur le fichier 'nom' les informations reçues de PRESTEL.	

			Page
	¦LOADPRESTEL,"nom"	Charge le fichier 'nom' dans le tampon PRESTEL.	
	¦CURSOR,col,lig	Impose la position du curseur.	
	¦REFRESH ¦REFRESH,lig	Rafraîchit l'écran PRESTEL. Rafraîchit une seule ligne.	

SOUS CP/M

**	STAT LST:=TTY:	Dirige la sortie imprimante vers le RS 232.
***	PIP dest=sour	Copie source vers destination source peut être un fichier ou CON: (le clavier) ou RDR: ou AUX: (le RS 232) dest peut être un fichier ou CON: (l'écran) ou PUN: ou AUX: (le RS 232).
2**	SETUP	Appelle un menu pour fixer les paramètres de transmission.
+**	SETSIO pam	Fixe les paramètres de transmission. Les paramètres sont : TX vitesse (vitesse d'émission) ; RX vitesse (vitesse de réception) ; BITS n (nombre de bits de donnée) ; PARITY NONE/ODD/EVEN (parité) ; STOP n (nombre de bits d'arrêt) ; HANDSHAKE ON/OFF ; XON ON/OFF.

La mémoire supplémentaire sur 6128

			Page
***	RUN "BANKMAN"	Active la gestion de la mémoire supplémentaire.	118
**	¦SCREENCOPY,n,a ¦SCREENCOPY,p,n,a	Copie le bloc écran a dans n. Comme ci-dessus, mais en faisant 64 étapes ; p est le n° d'étape.	118
**	¦SCREENSWAP,n,a ¦SCREENWAP,p,n,a	Échange les écrans n et a. Échange par étapes.	119
**	¦BANKOPEN,n	Ouvre un fichier virtuel de taille d'enregistrement n.	119
**	¦BANKWRITE, @R%,X$,E	Écrit X$ sur l'enregistrement E, code de retour R%.	119
**	¦BANKREAD, @R%,X$,E	Lit X$ sur l'enregistrement E, code de retour R%.	119
*	¦BANKFIND, @R%,X$,E1,E2	Recherche la chaîne X$.	120

Annexe 5
INDEX DES PROGRAMMES

CONSEILS DE LECTURE

Pour approfondir vos connaissances en BASIC Amstrad et mieux connaître le système des CPC 464, 664 et 6128, P.S.I. vous propose une palette d'ouvrages utiles.

POUR MAÎTRISER LE BASIC AMSTRAD

□ **BASIC Amstrad : 1 - Méthodes pratiques** – Jacques Boisgontier et Bruno Césard (Éditions du P.S.I.)

Pour ceux qui ont déjà pratiqué un BASIC, voici un ouvrage de perfectionnement au BASIC Amstrad. Un chapitre sur le CP/M 2.2 et le CP/M Plus donne les principales commandes systèmes.

□ **BASIC Amstrad : 2 - Programmes** – Jacques Boisgontier (Éditions du P.S.I.)

Pour pratiquer le BASIC Amstrad, cet ouvrage donne de nombreux programmes de gestion, d'éducation et de jeu où le rôle des fichiers est expliqué et largement commenté.

□ **BASIC plus 80 routines sur Amstrad** – Michel Martin (Éditions du P.S.I.)

Pour pousser votre Amstrad au maximum de ses capacités : 80 routines de simulation d'instructions qui n'existent pas en BASIC Amstrad.

POUR MIEUX CONNAÎTRE LE SYSTÈME DES CPC ET DU PCW 8256

□ **Clefs pour Amstrad** : **1 - Système de base** – Daniel Martin (Éditions du P.S.I.)

Mémento présentant synthétiquement le jeu d'instructions du Z80, les points d'entrée des routines système, les connecteurs et brochage, etc. Le livre de chevet du programmeur sur Amstrad.

□ **Clefs pour Amstrad** : **2 - Système disque** – Daniel Martin et Philippe Jadoul (Éditions du P.S.I.)

Ce deuxième tome consacré au système disque présente les points d'entrée des routines disque, les blocs de contrôle, la programmation et les brochages des circuits spécialisés... La deuxième partie du livre est aussi destinée aux possesseurs d'Amstrad 8256.

□ **Le livre de CP/M Plus sur Amstrad** – Yvon Dargery (Éditions du P.S.I.)

Toutes les commandes CP/M et CP/M Plus pour maîtriser le système des 6128 et 8256 : un ouvrage de référence illustré par de nombreux programmes.

□ **Le livre de l'Amstrad, tome 1** – Daniel Martin et Philippe Jadoul (BCM – Diffusé par P.S.I.)

Ce livre, destiné aux programmeurs des CPC 464 et 664, donne une étude complète de tous les circuits internes et analyse la structure interne du BASIC. Vous y trouverez, en outre, une étude complète des RSX et des programmmes de scrolling de traçage de rectangles, de coloriage de surface et de manipulation vectorielle.

POUR ÊTRE INFORMÉ RÉGULIÈREMENT DE L'ACTUALITÉ DES MICROS AMSTRAD

□ **MICROSTRAD,** revue bimestrielle du Groupe Tests

Pour exploiter au mieux les capacités de votre micro, vous trouverez au sommaire de chaque numéro un rendez-vous avec les rubriques clés :

– Découvrez la face cachée de votre CPC : astuces, idées, conseils, tout pour comprendre votre micro, son anatomie, son fonctionnement, sa programmation et exploiter ses capacités graphiques et sonores.

– Domptez votre CPC 464, 664 ou 6128 : passionnés, petits ou grands, spécialistes ou débutants, une information pratique et la compétence d'experts au service de votre micro.

– Programmez votre micro Amstrad : dans chaque numéro de MICROSTRAD, un cocktail de programmes (dessins, jeux, utilitaires, gestion, etc.) et des trucs de programmation.

Votre avis nous intéresse

- Pour nous permettre de faire de meilleurs livres, adressez-nous vos critiques sur le présent livre.
- Si vous souhaitez des éclaircissements techniques, écrivez-nous, nous adresserons votre demande à l'auteur qui ne manquera pas de vous répondre directement.

- Ce livre vous donne-t-il toute satisfaction?

____

____

____

____

____

- Y a-t-il un aspect du problème que vous auriez aimé voir abordé?

____

____

____

____

____

Comment avez-vous eu connaissance de ce livre?

- ☐ publicité
- ☐ catalogue
- ☐ boutique micro
- ☐ autres
- ☐ cadeau
- ☐ librairie
- ☐ exposition

Avez-vous déjà acquis des livres PSI?

lesquels? ____

____

qu'en pensez-vous? ____

____

____

Nom ____________________ Prénom ____________________ Age ________

Adresse ____

Profession ____

Centre d'intérêt ____

Achevé d'imprimer par l'imprimerie Tardy Quercy S.A. à Bourges
D. L. : Janvier 1986 - N° d'Imprimeur : 12720 - N° d'Édition : 86595-316-1



PÉRIPHÉRIQUES ET FICHIERS SUR AMSTRAD CPC 464, 664 et 6128

Cet ouvrage s'adresse à vous, possesseurs d'Amstrad 464, 664 ou 6128, qui souhaitez approfondir vos connaissances en Basic, en particulier en ce qui concerne la manipulation de fichiers et les périphériques.

Vous y étudierez en détail la gamme très complète des périphériques de vos CPC : lecteurs de cassette et de disquette, imprimantes connectables, crayon optique, manette de jeu et RS 232.

Vous apprendrez, enfin, à utiliser les disques en accès séquentiel à l'aide d'ordres Basic standard, et en accès direct à l'aide de routines originales.

Une introduction aux bases de données vous permettra de maîtriser la manipulation de fichiers dans vos applications.



ÉDITIONS DU P.S.I.
BP 86 - 77402 LAGNY S/MARNE CEDEX - FRANCE

ISBN 2-86595 - 316 - 5 **120 FF**

PHOTO L. MAUNOURY

PÉRIPHÉRIQUES ET FICHIERS SUR AMSTRAD CPC 464, 664 et 6128

P.S.I.



Document numérisé avec amour par :

AMSTRAD CPC

MÉMOIRE ÉCRITE

https://acpc.me/